

Learning Optimal Control of Synchronization in Networks of Coupled Oscillators using Genetic Programming-based Symbolic Regression

Synchronization control using GP

Julien Gout · Markus Quade · Kamran
Shafi · Robert K. Niven · Markus Abel

Received: date / Accepted: date

Abstract Networks of coupled dynamical systems provide a powerful way to model systems with enormously complex dynamics, such as the human brain. Control of synchronization in such networked systems has far reaching applications in many domains, including engineering and medicine. In this paper, we formulate the synchronization control in dynamical systems as an optimization problem and present a multi-objective genetic programming-based approach to infer optimal control functions that drive the system from a synchronized to a non-synchronized state and vice-versa. The genetic programming-based controller allows learning optimal control functions in an interpretable symbolic form. The effectiveness of the proposed approach is demonstrated in controlling synchronization in coupled oscillator systems linked in networks of increasing order complexity, ranging from a simple coupled oscillator system to a hierarchical network of coupled oscillators. The results show that the proposed method can learn highly-effective and interpretable control functions for such systems.

J. Gout, M. Quade, M. Abel
University of Potsdam
Karl-Liebknecht-Str. 24/25
14476 Potsdam, Germany and
Ambrosys GmbH
David-Gilly-Str. 1
14469 Potsdam, Germany
E-mail: julien.gout@uni-potsdam.de
E-mail: mquade@uni-potsdam.de
E-mail: markus.abel@physik.uni-potsdam.de

K. Shafi, R. K. Niven
School of Engineering and Information Technology
University of New South Wales
Canberra ACT 2600, Australia
E-mail: k.shafi@adfa.edu.au
E-mail: r.niven@adfa.edu.au

Keywords Dynamical systems · Synchronization control · Genetic programming

1 Introduction

The control of dynamical systems lies at the heart of modern engineering [1], and in many other disciplines, including physics [2, 3] and medicine [4, 5]. This paper specifically focuses on the control of synchronization in dynamical systems. Synchronization is a widespread phenomenon observed in many natural and engineered complex systems whereby locally interacting components of a complex system tend to coordinate and exhibit collective behavior [6, 7]. In dynamical systems, synchronization refers to the coordination phenomenon between multiple weakly-coupled independent oscillating systems that influences the overall dynamics of the system. The role of synchronization control is to moderate this behavior (e.g. to drive the system into or out of synch) by applying an external force or control signal [6]. Synchronization control has significant implications for numerous application domains in engineering and science, including communications [8], teleoperations [9] and brain modeling [10], to name a few. A more specialized overview of synchronization in oscillators, and especially phase oscillators, can be found in [11].

Several approaches exist for the control of dynamical systems, such as those based on control theory [12], mathematical and numerical optimization [13] and computational intelligence [14] techniques. The “optimal control” methods [15], in particular, aim at driving and maintaining a dynamical system in a desired state. This is generally achieved by finding a control law, in the form of a set of differential equations, which optimizes (by maximizing or minimizing) a cost function related to the control task. For instance, in a medical application, the control of body tremors (e.g., due to seizures) may be achieved by minimizing the amplitude of body oscillations.

If the system is known in terms of a mathematical description, linear theory can be used [16, 17] in many cases to find the optimal control. However, for nonlinear, extended and consequently complex systems, linear theory may fail. In such cases more general methods are needed to learn effective control laws. In this paper, we present our approach to infer control laws for complex dynamical systems using an evolutionary machine learning method. Specifically, we describe the application of genetic programming (GP), a well-known evolutionary algorithm, to control synchronization in coupled networks, including a hierarchical network of coupled oscillators. Unlike neural networks and other black-box artificial intelligence methods that are commonly applied to optimal control, GP allows dynamically learning complex control laws in an interpretable symbolic form – a method that is referred as symbolic regression [18, 19, 20]. In contrast to the earlier application of GP to control of dynamical systems [1, 21], we use a multi-objective formulation of GP, which allows learning relatively much sparser as well as multiple Pareto (or non-dominated) solutions [18, 19, 20]. The Pareto solutions can be further analyzed for insight

using the conventional analytical methods, such as bifurcation analysis – subject of our ongoing work.

We demonstrate the effectiveness of the proposed control approach through application to different dynamical systems with growing level of complexity, ranging from a single oscillator to a hierarchical network. For each system we show the useful optimal control terms found through symbolic regression using the GP algorithm to synchronize or desynchronize the oscillators. The application to the control of synchronization in a hierarchical network of oscillators is motivated, especially, by brain disorder problems in the medical domain. Body tremors occur when firing neurons synchronize in regions of brain [4]. In a normal brain state, neurons are coupled to neighboring neurons such that adjacent neurons influence mutually. If the firing is periodic, which may appear due to the inherent dynamics of the excitable neurons, this mutual influence may give rise to synchronization [6, 22]. If the coupling term is very large, this synchronization may extend over a whole region in our brain and thus over many neurons. Eventually this collective firing leads to shaky movements of hands, arms or the head, and is treated as a brain disorder. One remedy to this problem is to implant a control device which resets the neurons and counteracts the collective synchronization. An evident question then is how to design such a controller which also minimizes design cost, energy consumption, or other medical constraints. We model this phenomenon, in this work, as a set of oscillatory units (representing neurons) coupled in a small hierarchical network, and apply the proposed approach for learning the optimal control laws for this model.

To the best of our knowledge this is the first application of a multi-objective GP to synchronization control in networks of coupled dynamical systems.

The rest of the paper is organized as follows: Section 2 provides the background information on control of dynamical systems, the common approaches for optimal control of dynamical systems, GP, and the specific methods used in our implementation of GP. Section 2.2.3 provides the details of the common GP parameters and other experimental settings used to evaluate the proposed method. The specific parameter settings are provided in each corresponding section. As a proof of concept, Section 3 demonstrates the application of GP based control using two simple benchmark dynamical system examples: a harmonic oscillator and the Lorentz system. GP is used to learn control to bring these system into a chaotic state or back. Section 4 presents our study of GP application to networked dynamic systems. Four systems are tested in this section, including a simple coupled oscillator system; and three systems of oscillators coupled, respectively, in a one-dimensional ring network; a two-dimensional torus network; and a hierarchical network. The paper concludes in Section 5.

2 Methods

2.1 Optimal control of dynamical systems

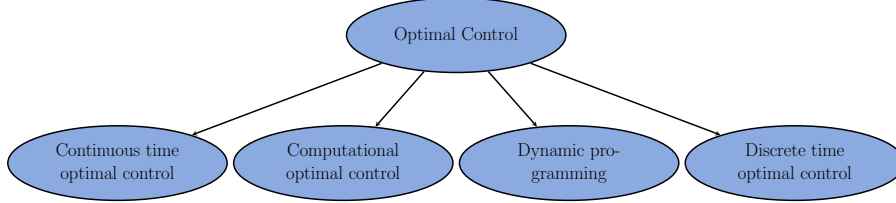


Fig. 1 An overview of different types of optimal control.

The control of a dynamical system generally involves determining and manipulating system's state trajectories over a spatio-temporal regime in order to drive the system to a desired state. The control problem can be formulated as an optimization task with the objective to minimize a cost or error function defined in terms of system's deviation from its desired state over the state space of interest. In general, a formal definition of an optimal control problem requires: a mathematical model of the system and its dynamics; a performance index or cost function; a specification of all boundary conditions on states; and constraints to be satisfied during the control.

If there are no path constraints on the states or the control variables, and if the initial and final conditions are fixed, a fairly general continuous time optimal control problem may be defined as follows: Find the control vector $\mathbf{u} : \mathbb{R}^{n_x} \times [t_s, t_f] \mapsto \mathbb{R}^{n_u}$ to minimize the performance index

$$\Gamma = \varphi(\mathbf{x}(t_f)) + \int_{t_s}^{t_f} L(\mathbf{x}(t), \mathbf{u}(\mathbf{x}, t), t) dt, \quad (1)$$

subject to

$$\dot{\mathbf{x}} = \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}, t), \quad \mathbf{x}(t_s) = \mathbf{x}_s, \quad (2)$$

where $[t_s, t_f]$ is the time interval of interest; $\mathbf{x} : [t_s, t_f] \mapsto \mathbb{R}^{n_x}$ is the state vector; $\varphi : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ is a terminal cost function; $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \mapsto \mathbb{R}$ is an intermediate cost function; and $\tilde{\mathbf{f}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \mapsto \mathbb{R}^{n_x}$ is a vector field. Note that Equation (2) represents the dynamics of the system and its initial state. This problem definition is known as the Bolza problem; and for $\varphi(\mathbf{x}(t_f)) = 0$ and $\mathbf{u} = \dot{\mathbf{x}}(t)$ it is known as the Lagrange problem. Also note that the performance index Γ is a functional, which is used to assign a real value to each control function \mathbf{u} in a class.

The solutions to many optimal control problems cannot be found by analytical means. Over the years, many computational methods have been developed to solve general optimal control problems. The choice of a method for

addressing an optimal control problem may depend on a number of factors, including the types of cost functions, time domain, and constraints considered in the problem. Figure 1 shows different methods used in the optimal control of dynamical systems. Among these methods, the direct methods work by discretizing the control problem and solving it using non-linear programming approaches. Some methods involve the discretization of the differential equations by defining a grid of N points covering the time interval $[t_s, t_f]$, $t_s = t_1 < t_2 < \dots < t_N = t_f$, and solving these equations using, for instance, Euler, Trapezoidal, or Runge-Kutta methods. In this approach, the differential equations become equality constraints of the nonlinear programming problem. Other direct methods involve the approximation of control and states using basis functions, such as splines or Lagrange polynomials.

The continuous-time problems mentioned above have discrete-time counterparts. These formulations are useful when the dynamics are discrete (for example, a multistage system), or when dealing with computer controlled systems. In discrete-time, the dynamics can be expressed as a difference equation:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), k), \quad \mathbf{x}(N_0) = \mathbf{x}_s,$$

where k is an integer index, $\mathbf{x}(k)$ is the state vector, $\mathbf{u}(k)$ is the control vector, and \mathbf{f} is a vector function. The objective is to find a control sequence $\mathbf{u}(k)$, $k = N_0, \dots, N_f - 1$, to minimize a performance index of the form:

$$\Gamma = \varphi(\mathbf{x}(N_f)) + \sum_{k=N_0}^{N_f-1} L(\mathbf{x}(k), \mathbf{u}(k), k).$$

See, for example, [23] and [24] for further details on discrete time optimal control.

Dynamic programming is an alternative to the variational approach to optimal control. It was proposed by Bellman in the 1950s, and is an extension of Hamilton-Jacobi theory. A number of books exist on these topics including [23], [16], and [24]. A general overview of the optimal control methods for dynamical systems can be found in [15]. For further details, readers are referred to [25].

Unless otherwise stated, we approach the control problems presented in this paper using discrete-time numerical methods. However for most purposes, the continuous-time formulation given in Equations (1) and (2) can be adopted unchanged for our control methods. One major generalization, though, will be made in the context of multi-objective optimization (Section 2.2.1); there Γ is replaced by a vector of independent cost functionals $\mathbf{\Gamma} = (\Gamma_1, \dots, \Gamma_N)$. Another adjustment concerns the specialization of \mathbf{f} and \mathbf{u} for the particular control scheme considered here, as will be described next.

A feedback control scheme [26] is adopted in this work to implement the control. Figure 2 depicts the general architecture.

For this particular scheme Equation (2) can be rewritten as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \mathbf{a}, \quad \mathbf{x}(t_s) = \mathbf{x}_s,$$

where the uncontrolled system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ is controlled by an additive actuator term \mathbf{a} , and the control function now depends on sensor measurements, given by the output vector $\mathbf{s} \in \mathbb{R}^{n_s}$:

$$\mathbf{a} = \mathbf{u}(\mathbf{s}, t).$$

These measurements might be nonlinear functions of the state \mathbf{x} . For simplicity external perturbations to the dynamic system are not considered here.

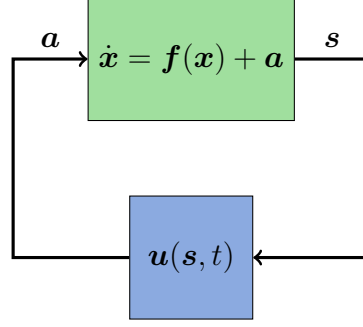


Fig. 2 Sketch of the feedback control loop. The output of the dynamical system \mathbf{x} is measured by sensors \mathbf{s} which are used as input to the control function \mathbf{u} . The control function, in turn, acts on the system via actuators $\mathbf{a} = \mathbf{u}(\mathbf{s}, t)$ in order to achieve a desired state. (External disturbances which can be incorporated explicitly as additional inputs to the dynamical system and the control function are not shown here.)

2.2 Genetic Programming

Genetic Programming (GP) [27,28] is a well-known evolutionary algorithm which falls under the general class of meta-heuristic search techniques that promise global optimization. Similar to the infamous genetic algorithm (GA), GP also uses the natural selection metaphor to evolve a set, or so-called population, of solutions or individuals using a fitness-based selection mechanism. The evolution occurs over a number of iterations, called generations. GP differs from GA mainly in the solution representation. A solution in GP is generally represented using lists or expression trees. Expression trees are constructed from the elements of two predefined primitive sets: a function set consisting of mathematical operators and trigonometric functions, such as $\{+, -, *, \cos, \sin\}$; and a terminal set consisting of variables and constants, such as $\{x, y, b\}$. Function symbols make up the internal nodes of a tree; and terminal symbols are used in the leaf nodes. For example, Figure 3 shows the tree representation for the expression $b \cdot x + \cos(y)$. All elements of the tree are drawn from the aforementioned primitive sets: the variables and constants in the terminal set (x , y , and b) form the leaves of the tree and the mathematical

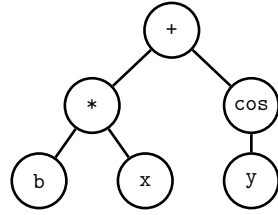


Fig. 3 Tree representation of the mathematical expression $b \cdot x + \cos(y)$. The symbols b , x , and y are taken from the terminal set and make up the leaf nodes of the tree, whereas the symbols $*$, $+$, and \cos , are symbols taken from the function set, they make up the internal nodes.

symbols in the functional set (\cdot , $+$, and \cos) are used in forming the tree's internal nodes.

A standard GP algorithm begins by generating a population of random solutions. A random variable length solution, with a given maximum tree depth, is generated by choosing operators, functions and variables from the two sets uniform randomly. Each solution is then evaluated in a given task, e.g. learning underlying relationship between a given set of variables. A fitness value is assigned to each solution based on its performance in solving the task, e.g. how closely a solution predicted the target function output. A new population of solutions is then generated by: (i) probabilistically selecting parent solutions from the existing population using a fitness-proportionate selection mechanism; and (ii) creating offspring or new solutions by applying recombination (or crossover) and variation (or mutation) operators (see Figure 4). This operation is repeated until a given number of solutions (a fixed population size) is reached. A closure property is always maintained to ensure only valid solutions are generated during both the initialization and breeding operations. An elitist approach is commonly used for improving the algorithm's convergence speed. This involves copying some of the high performing parent solutions to the next generation population. The selection, evaluation and reproduction processes are repeated until a given stopping criteria is met, commonly a fixed number of maximum generations. For further details about GP operation, readers are referred to [29, 30].

The original motivation behind GP was to automatically produce computer programs that can perform useful tasks similar to human developed computer programs [27]. However since its inception, GP has been applied to various tasks, including the traditional machine learning [31] and optimization [14] tasks. In the context of learning dynamical system models or control laws for dynamical systems, GP is a preferred choice for two main reasons: First, the expression tree representation used by GP is human interpretable and clearly provides an edge over the blackbox models learnt by other computational approaches, such as artificial neural networks. Second, the GP solutions can be readily represented as mathematical equations and evaluated as control laws for dynamical systems.

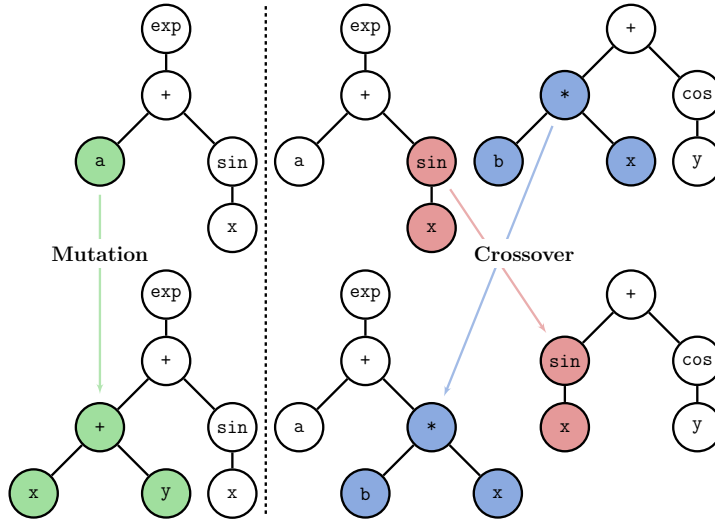


Fig. 4 Breeding: The mutation and crossover operations on expression trees. (Source: Adapted from [20]; used with permission.)

The application of GP to the general control problem of concern, described in Section 2.1, can be formulated as a learning and optimization task. That is, we would like to learn a control function to drive and keep a dynamical system, e.g. a harmonic oscillator, in a desired state. This requires minimizing a cost or objective function (\mathbf{J}), e.g. the difference between a given state in time and the desired state. For most practical purposes, \mathbf{J} can be expected to have complex properties including non-linearity, multi-modality, multivariate and discontinuity. This poses a serious challenge to many traditional direct and gradient methods, making heuristic methods, such as the GP, suitable candidates to be applied to this task. Figure 5 depicts how GP-based dynamic controller integrates within the general feedback control framework shown in Figure 2.

2.2.1 Multi-Objective Fitness Evaluation

Recalling our discussion above, fitness assignment is a key process in GP, and in any evolutionary algorithm for that matter, which determines the quality of a solution in the population. A common method for fitness assignment is to map solutions' performances to scalar values within a given range, e.g. $[0, 1]$. This method simplifies the ranking procedure needed for selection and reproduction processes. However, it also limits the number of performance objectives that can be considered in the fitness evaluation. A straight-forward way to address this concern is to apply a weighted sum method that in turn still allows mapping multiple performance objectives to scalar values. This type of methods not only require manual tuning of weights but also hide trade-off details between conflicting objectives. An alternative method to handle conflicting

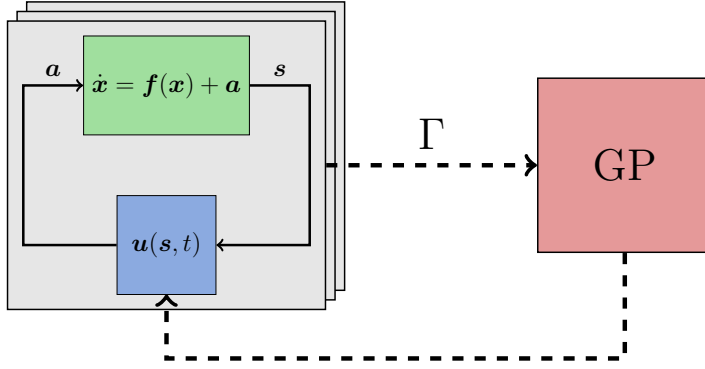


Fig. 5 Sketch of the machine learning loop. Using an evolutionary metaphor, the GP algorithm generates a set of candidate control solutions \mathbf{u} , called the population. The candidate solutions are then evaluated in many realizations of the control loop; the performance in each iteration is rated via a cost functional Γ and fed back as a cost index into the GP algorithm. The algorithm uses the performance rating to select the best solutions and evolve them into the next generation of candidate solutions. This learning loop repeats until at least one satisfying control law is found (or other break conditions are met).

performance objectives in the design of fitness functions is to use the concept of Pareto dominance [32]. According to this principle a solution x dominates another solution y , if x performs better than y in at least one of the multiple objectives or criteria considered and at least equal or better in all other objectives. This concept provides a convenient mechanism to consider multiple conflicting performance objectives simultaneously in ranking solutions based on their domination score. The solutions with the highest scores form the Pareto or efficient frontier. Figure 6 provides an illustration of this concept. Several very successful evolutionary multi-objective optimization (EMO) algorithms are based on Pareto dominance [33, 34, 35].

The standard GP is known to have a tendency to generate long and complicated solutions in order to exactly match, or overfit, performance target (optimal performance in our case). One way to address this issue is to design a fitness function where both the performance (e.g., controller error in our case) and length of solutions are considered explicitly in determining the quality of a solution. Such a multi-objective fitness mechanism allows introducing an explicit selection pressure in the evolutionary process and preferring smaller well-performing solutions over their longer counterparts. Following this line, a multi-objective fitness evaluation method is adopted in our implementation of GP in this work. In specific, we adopt the mechanism used in NSGA-II [33] in our implementation of GP, which combines a non-dominated sorting mechanism with an Euclidean distance-based metric to promote solution diversity and spread or coverage of the entire Pareto front.

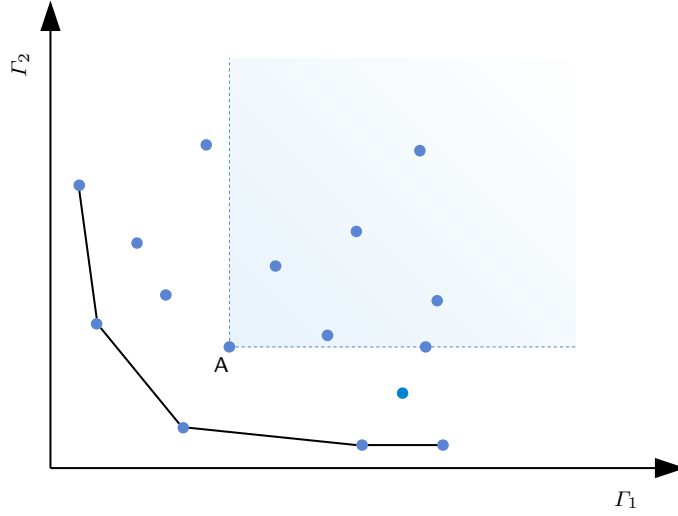


Fig. 6 Pareto front (blue dots connected by a black line) of a set of candidate solutions evaluated with respect to two cost indexes Γ_1 and Γ_2 . The Pareto-optimal solutions are non-dominated with respect to each other but dominate all other solutions. The region marked in light blue illustrates the notion of Pareto dominance: solutions contained within that region are dominated by a solution A.

2.2.2 Constant Optimization

The learning of numeral constants, if desired, is treated similar to the learning of other terminal variables in the standard GP and an approximation may be learned by sampling over a given range. However, this approach can severely impair the convergence speed of GP as the search space essentially becomes infinite, especially for a continuous representation. Another approach, followed in this work, is to use a traditional parameter optimization algorithm, like the Levenberg-Marquardt least squares method. By allowing designated symbolic constants $\mathbf{k} = (k_1, k_2, \dots)$ in an expression tree, one can determine optimal values for these constants through parameter optimization. Thus, the calculation of the numeral constants becomes another optimization task, with

$$\mathbf{k}^* = \underset{\mathbf{k}}{\operatorname{argmin}} \Gamma(\mathbf{u}(\mathbf{s}, \mathbf{k})),$$

and

$$\Gamma = \Gamma(\mathbf{u}(\mathbf{s}, \mathbf{k}^*)),$$

effectively introducing two combined layers of optimization.

To incorporate such a regression mechanism, the terminal set can be divided further into an argument set and a constant set. Where the argument set contains all the terminal symbols that represent elements from the sensor vector \mathbf{s} , which are passed as arguments to the control function \mathbf{u} . The constant set, on the other hand, consists of all the designated constants representing elements from the constant vector \mathbf{k} . The actual construction of these sets depends on the dynamical system under consideration and the type of the control task.

2.2.3 GP setup

This section gives a brief summary of the specific implementation details and the common parameters used in our experimental set up. Problem-specific methods and parameters are stated in the respective sections.

All computer programs are developed in Python using open source software packages. The implementation of the GP algorithm uses a customized version of the *deap* module (Distributed Evolutionary Algorithms in Python) [36]. Particularly the routines for tree generation, selection, and breeding were adopted unchanged. Most of the numerical algorithms in use are provided by the *numpy* and *scipy* modules [37,38]. Notably, constant optimization is conducted using the Levenberg-Marquardt least squares algorithm (*scipy*) and numerical integration using the *dopri5* solver (also *scipy*). Random numbers are generated using the Mersenne Twister pseudo-random number generator provided by the *random* module [39]. Finally, the *sympy* module is used for the simplification of symbolic mathematical expressions generated from the GP runs [40]. The code can be found at <https://github.com/ambrosys/glyph>.

Table 1 gives an overview of the methods and parameters used for the GP runs. Actual implementations can be found under the same name in the *deap* module.

The function set is chosen such that the operators and functions are defined on the whole of \mathbb{R} . This allows for an easy evaluation and application of the expressions built from this set and prevents additional handling of corner cases, like singularities. This choice, though, puts a restriction on the number of possible solutions. In principle, the inclusion of other functions and operators, such as \log , $\sqrt{\cdot}$, or $1/x$, is conceivable and would lead to a different space of potential solutions. Where possible, other GP parameter values are chosen according to the best practices used by the GP community, see [27,31,29]. A relatively smaller population size and lower maximum number of generations is used to keep the computational cost within the allocated bounds of this project. In addition, for more involved cases a second stopping criterion is used where the learning is stopped when an error of the order of 10^{-5} is reached. Numerical integration is performed excessively during cost assessment for all of the investigated dynamical systems. As mentioned before, *dopri5* is used as solver: This is an explicit Runge-Kutta method of order (4)5 with adaptive step size. If not otherwise stated, the maximum number of steps allowed during one call to the integrator defaults to 4000, the relative tolerance to 10^{-6} ,

Table 1 General setup of the GP runs.

function set	$\{+, -, \cdot, \sin, \cos, \exp\}$
population size	500
max. generations	20
MOO algorithm	NSGA-II
tree generation	<i>halfandhalf</i>
min. height	1
max. height	4
selection	<i>selTournament</i>
tournament size	2
breeding	<i>varOr</i>
recombination	<i>cxOnePoint</i>
crossover probability	0.5
crossover max. height	20
mutation	<i>mutUniform</i>
mutation probability	0.2
mutation max. height	20
constant optimization	<i>leastsq</i>

and the absolute tolerance to 10^{-12} . The pseudo-random number generator is seeded by a randomly selected unique seed for every GP run and never reinitialized during the same run. The specific seed used in an experiment will be explicitly stated in the corresponding setup description. Results obtained from one experiment can thus be duplicated when this same seed is reused in another run of the experiment.

3 Control of independent dynamical systems

In order to establish a baseline, explain the working of the proposed method and determine its effectiveness, in this section we discuss the application of GP-based control methodology to two prominent and well-understood examples: the harmonic oscillator; and the Lorenz system [41]. Both systems are forced to a fixed point and the resulting control laws are analyzed in detail.

3.1 Harmonic Oscillator

Consider a harmonic oscillator incorporated into the feedback control scheme discussed in Section 2.1. The dynamical system can be stated as:

$$\ddot{x} = -\omega_0^2 x + u(x, \dot{x}), \quad (3)$$

with the particular system parameters defined in the left half of Table 2. Ideal sensors are assumed to measure position and velocity, $\mathbf{s} = (x, \dot{x})$, thus, the explicit statement of the sensor function is omitted in the argument list of u .

The control target is to drive the harmonic oscillator toward a steady state resting position. This may be formulated into a cost function using the root mean squared error (RMSE) of the trajectory x with reference to 0, that is,

$$\Gamma_1 := \text{RMSE}(x, 0) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x(t_i) - 0)^2}.$$

Since this is a numerical experiment the RMSE is formulated in a discrete form, with time steps $t_i = i \frac{T}{N}$ ($i = 0, \dots, N-1$), and an oscillation period $T = \frac{2\pi}{\omega_0}$. A time interval of twenty periods, as defined in Table 2, is large enough to get a meaningful measurement of Γ_1 . The number of discrete time steps, n , is chosen such that an accuracy of 50 samples per period is achieved.

Further, as discussed in Section 2.2.1, the expression length is used as the second objective to bias GP learning towards smaller control laws:

$$\Gamma_2 := \text{length}(u),$$

which, in this case, simply corresponds to a count of nodes in the expression tree for u .

Table 2 Harmonic oscillator system setup.

dynamic system		GP	
ω_0	$\exp(2)$	cost functionals	$\text{RMSE}(x, 0)$
$x(t_0)$	$\ln(4)$		$\text{length}(u)$
$\dot{x}(t_0)$	0	argument set	$\{x, \dot{x}\}$
t_0, t_n	$0, 20 \frac{2\pi}{\omega_0}$	constant set	$\{k\}$
n	1000	seed	1730327932332863820

These two objective or cost functions are listed as part of the GP setup in the right half of Table 2. Additionally, the primitives from the argument set and constant set are specified. In this case the argument set consists of symbols representing position and velocity, the two quantities measured by \mathbf{s} ; the constant set consists of a single constant, k , that is used to perform constant optimization. For easier readability the notation does not distinguish between primitive symbols and variable names, that is, x is used instead of \mathbf{x} , and \dot{x} instead of $\mathbf{x_dot}$. The seed from Table 2 is used to initialize the pseudo-random number generator of the GP algorithm and leads to the particular solutions presented next. Other relevant parameters are stated in the general GP setup, as described in Section 2.2.3.

The Pareto solutions for this particular setup are presented in Table 3 in ascending order sorted by Γ_1 (RMSE). Not surprisingly, more complex expressions tend to provide better control (in terms of lower NRMSE). One can also observe multiple mathematically equivalent solutions in the Pareto set (e.g., the two expressions of length 6). Although equivalent, these expressions are

distinct as far as the internal representation is concerned and the GP algorithm treats them as independent solutions.

Table 3 Control of the harmonic oscillator: Pareto-front solutions.

RMSE	length	expression	constants
0.043973	10	$k \cdot (-k \cdot x + x - \dot{x})$	$k = 151.907120$
0.043976	8	$k \cdot (-k \cdot x - \dot{x})$	$k = 151.232316$
0.115862	7	$\exp(-k \cdot x - \dot{x})$	$k = 3509.921747$
0.123309	6	$k \cdot (-x - \dot{x})$	$k = 8.545559$
0.123309	6	$-k \cdot (x + \dot{x})$	$k = 8.545559$
0.123309	5	$k \cdot (x + \dot{x})$	$k = -8.545254$
0.127432	3	$k \cdot \dot{x}$	$k = -7.389051$
0.241743	2	$-\dot{x}$	
0.801177	1	k	$k = 25.229759$

Figure 7 shows the trajectories of two particular solutions from Table 3 (first row and the third row from the bottom). Both look like underdamped cases of the damped harmonic oscillator system. We will analyze both solutions in more detail.

First consider $u(x, \dot{x}) = k(-kx + x - \dot{x})$. Inserting into the general Equation (3) for the controlled harmonic oscillator one gets

$$\begin{aligned}
 \ddot{x} &= -\omega_0^2 x + k(-kx + x - \dot{x}) \\
 &= -(\omega_0^2 + k^2 - k)x - k\dot{x} \\
 &= -\tilde{\omega}_0^2 x - k\dot{x}
 \end{aligned} \tag{4}$$

with $\tilde{\omega}_0^2 := \omega_0^2 + k^2 - k$. This is indeed the differential equation for the damped harmonic oscillator. Since $\omega_0^2 > 1$, it follows, that $\tilde{\omega}_0^2 > 0$ and the condition for the underdamped case, $\frac{k^2}{4} < \tilde{\omega}_0^2$, is fulfilled for any $k \in \mathbb{R}$. Using the initial values from Table 2, we get the particular solution

$$x(t) = e^{-\frac{k}{2}t+2} \left(\cos(\omega t) + \frac{k \sin(\omega t)}{2\omega} \right), \tag{5}$$

where $\omega^2 := \tilde{\omega}_0^2 - \frac{k^2}{4}$.

The form of solution (5) demands an answer to the specific value found for k : One would expect large values of $k \gg 151.9$ (up to the numeric floating point limit), since one of the goals of optimization is to drive the harmonic oscillator to zero, and the particular solution (5) implies that $x(t) \rightarrow 0$ as $k \rightarrow \infty$ ($t \in \mathbb{R}$). Why the least squares algorithm finds a considerably smaller value can be explained by the choice of discretization made here. Figure 8 illustrates how the optimal value k^* depends on the discretization of the finite time interval, more specifically the step size Δt . Since the result of numerical integration is restricted by the resolution of the time interval, starting at the

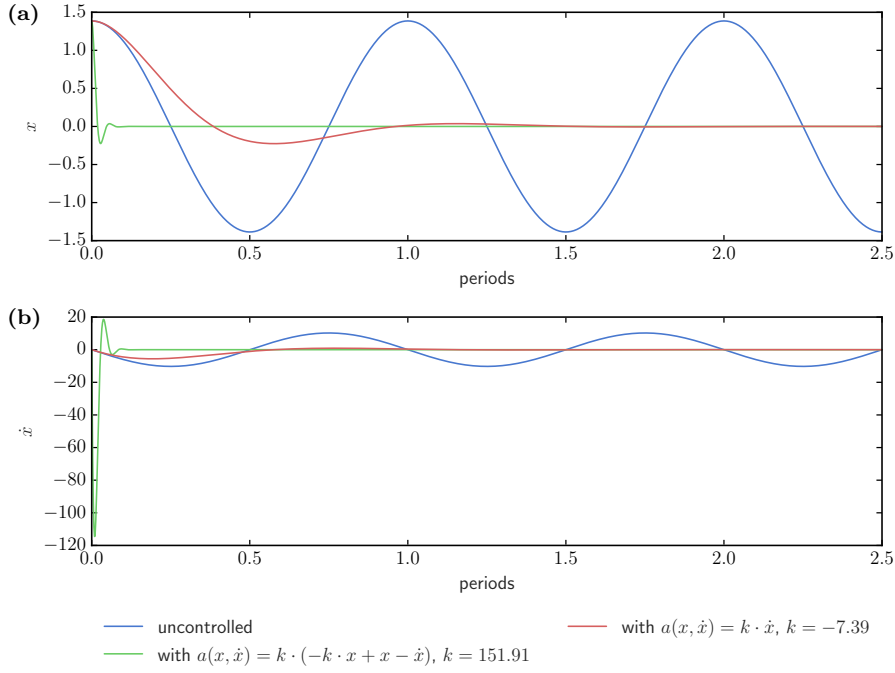


Fig. 7 Control of the harmonic oscillator. The trajectories of two candidate solutions, chosen from the Pareto front, that drive the harmonic oscillator to zero are shown. The best solution regarding Γ_1 are shown in a green color. A simple yet moderately good solution with respect to Γ_1 is shown in red. For reference the uncontrolled system is shown in blue. **a:** Position, **b:** speed of the oscillators.

initial position $x(0) = e^2$, the shortest time span possible for the trajectory to reach zero is Δt . Thus, setting an upper bound k^* when optimizing for the RMSE of the whole trajectory with respect to zero: values larger k^* would not improve the cost index any further.

As the second case consider the solution $u(x, \dot{x}) = k \cdot \dot{x}$. Again inserting into (3)

$$\ddot{x} = -\omega_0^2 x + k\dot{x}, \quad (6)$$

one gets a damped harmonic oscillator system. The difference to the previous case (5) being, that the coefficient of x does not depend on k . This allows for a wider range of solutions that cover all regimes of the damped harmonic oscillator (i.e. overdamped, underdamped, and diverging case). A numerical analysis of the RMSE with respect to k shows the presence of a single minimum in the underdamped regime, see Figure 9. The result $k^* = -\omega_0$ from constant optimization corresponds almost exactly to the minimum position, as would be expected.

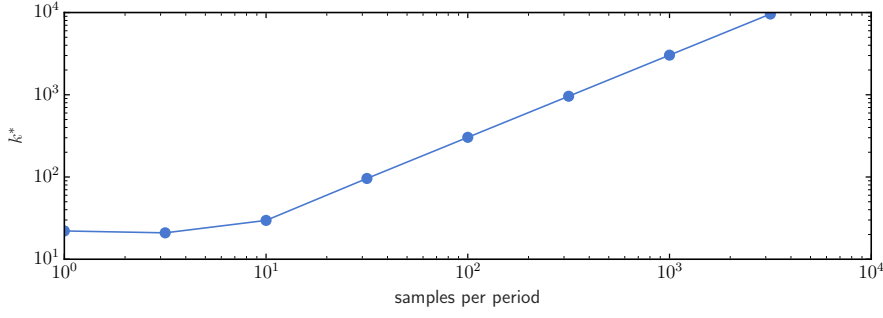


Fig. 8 Control of the harmonic oscillator. Optimal parameter k^* for the control law $u(x, \dot{x}) = k \cdot (-k \cdot x + x - \dot{x})$ as determined by the least squares optimization using different sampling rates f_s . Starting at $f_s = 10$ the optimal value for k increases linearly with the sampling rate, $k^* \sim f_s \sim 1/\Delta t$.

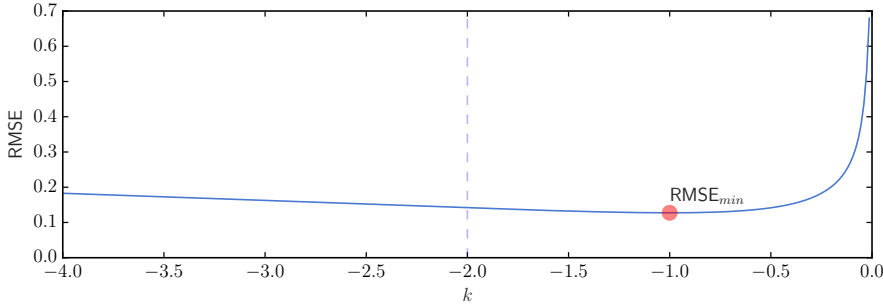


Fig. 9 Control of the harmonic oscillator. Minimum in RMSE with respect to k for the control law $u(x, \dot{x}) = -k \cdot \dot{x}$. The minimum, at $k = -7.389 \approx -\omega_0$, lies in the underdamped regime. Solutions in the overdamped regime, left of the aperiodic borderline case (dashed line), result in strictly increasing RMSE as $k \rightarrow -\infty$. So do the diverging solutions for $k > 0$, as $k \rightarrow \infty$ (not shown).

3.2 Lorenz System

While the harmonic oscillator is rather a simple case to solve, the control of a well known non-linear system, the Lorenz system, is studied in this section. Such systems are much harder to predict and control in practice due to their chaotic behavior in that small changes in the initial state may lead to exponential divergence of trajectories. The Lorenz model [41] consists of a system of three ordinary differential equations:

$$\begin{aligned} \dot{x} &= s(y - x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= xy - bz, \end{aligned} \tag{7}$$

with two nonlinearities, xy and xz . Here x , y , and z make up the system state and s , r , b are parameters: s is the Prandtl number, r is the Rayleigh number,

and b is related to the aspect ratio of the air rolls. For a certain choice of parameters and initial conditions chaotic behavior emerges.

Here we present two examples where the target is to learn control of bring a chaotic Lorenz system to a complete stop, that is, $(x, y, z) = 0$ ($t \in \mathbb{R}$). In the first example, the actuator term is applied to \dot{y} . This allows for a more direct control of the system, since y appears in every equation of (7) and, thus, influence all three state components, x , y , and z . In the second example the actuator term is applied to \dot{z} , which leads to a more indirect control, since the flow of information from z to x is only through y .

Table 4 Control of the Lorenz system: system setup.

dynamic system		GP	
s	10	cost functionals	RMSE($x, 0$)
r	28		RMSE($y, 0$)
b	8/3		RMSE($z, 0$)
$x(t_0)$	10.0		length(u)
$y(t_0)$	1.0	argument set	$\{x, y, z\}$
$z(t_0)$	5.0	constant set	$\{k\}$
t_0, t_n	0, 100	seed (in y)	4360036820278701581
n	5000	seed (in z)	2480329230996732981

Like the harmonic oscillator case, the system setup is summarized in Table 4. When $r = 28$, $s = 10$, and $b = 8/3$, the Lorenz system produces chaotic solutions (not all solutions are chaotic). Almost all initial points will tend to an invariant set – the Lorenz attractor – a strange attractor and a fractal. When plotted the chaotic trajectory of the Lorenz system resembles a butterfly or figure eight (blue graph in Figure 10). The target of control is, again, formulated as RMSE of the system state with respect to zero (separately for each component)

$$\Gamma_1 := \text{RMSE}(x, 0), \Gamma_2 := \text{RMSE}(y, 0), \Gamma_3 := \text{RMSE}(z, 0).$$

The control function u can make use of ideal measurements of the state components. Constant optimization is performed on a single constant k . The respective GP runs for control in y and control in z are conducted with the corresponding random seeds labeled “in y ” and “in z ”.

Control in y : For control in y the actuator term u is added to the left side of the equation for \dot{y} in the uncontrolled system (7):

$$\dot{y} = rx - y - xz + u(x, y, z).$$

The Pareto solutions from the GP run are shown in Table 5. The wide spread of the cost indices is a sign of conflicting objectives that are hard to satisfy in conjunction. Interestingly, almost all solutions, u , commonly introduce a negative growth rate into \dot{y} . This effectively drives y to zero and suppresses

the growth terms, sy and xy , in the equations for \dot{x} and \dot{z} respectively, in turn, driving x and z to zero as well. As would be expected, minimal expressions, of length 1 or 2, cannot compete in terms of the RMSE, but, the correlation between expression length and RMSE is not as strong as for the harmonic oscillator in Section 3.1. For example, the simple solution, $u(x, y, z) = -ky$ (fourth row), is almost as good as the lengthier one, $u(x, y, z) = -\exp(x) + ky$ (first row), and even better in RMSE_y .

Table 5 Control of the Lorentz system in y : Pareto-front solutions.

RMSE_x	RMSE_y	RMSE_z	length	expression	constants
0.178884	0.087476	0.105256	7	$-\exp(x) + k \cdot y$	$k = -135.43$
0.241226	0.069896	0.213063	5	$k \cdot x + z$	$k = -27.84$
0.246315	0.014142	0.222345	6	$-z + k \cdot y$	$k = -75590.65$
0.246316	0.014142	0.222347	4	$-k \cdot y$	$k = 75608.50$
0.246367	0.028851	0.220426	10	$-x \cdot (k + y) \cdot \exp(\exp(y))$	$k = 9.62$
0.246729	0.118439	0.211212	6	$-x \cdot (k + y)$	$k = 29.21$
0.246850	0.031747	0.220726	9	$-x \cdot (k + y) \cdot \exp(y)$	$k = 26.12$
4.476902	4.468534	7.488516	3	$-\exp(y)$	
7.783655	8.820086	24.122441	2	$-x$	
7.931978	9.066296	25.047630	1	k	$k = 1.0$
8.319191	8.371462	25.932887	2	$-y$	
8.994685	9.042226	30.300641	1	z	

Table 5 shows the results from the GP run. One solution immediately stands out: $u = k \cdot x + z$, with $k = -27.84$ (second row). It is exactly what one might expect as a control term for the chaotic Lorenz system with control in y . This control law effectively reduces the Rayleigh number r to a value close to zero ($k \approx r$), pushing the Lorenz system past the first pitchfork bifurcation, at $r = 1$, back into the stable-origin regime. If $r < 1$ then there is only one equilibrium point, which is at the origin. This point corresponds to no convection. All orbits converge to the origin, which is a global attractor, when $r < 1$.

The phase portrait of the solution from the first and second row of Table 5 are illustrated in Figure 10. After a short excursion in negative y direction ($t \approx 5$), the green trajectory quickly converges to zero. The red trajectory seems to take a shorter path in phase space, but, it is actually slower to converge to the origin. This is verified by a plot of the trajectories for the separate dimensions x , y and z over time Figure 11.

Control in z : For control in z the actuator term u is added to the left side of the equation for \dot{z} in the uncontrolled system (7)

$$\dot{z} = xy - bz + u(x, y, z)$$

Selected Pareto-front individuals from the GP run are displayed in Table 6. As mentioned at the beginning of this section, effective control is hindered

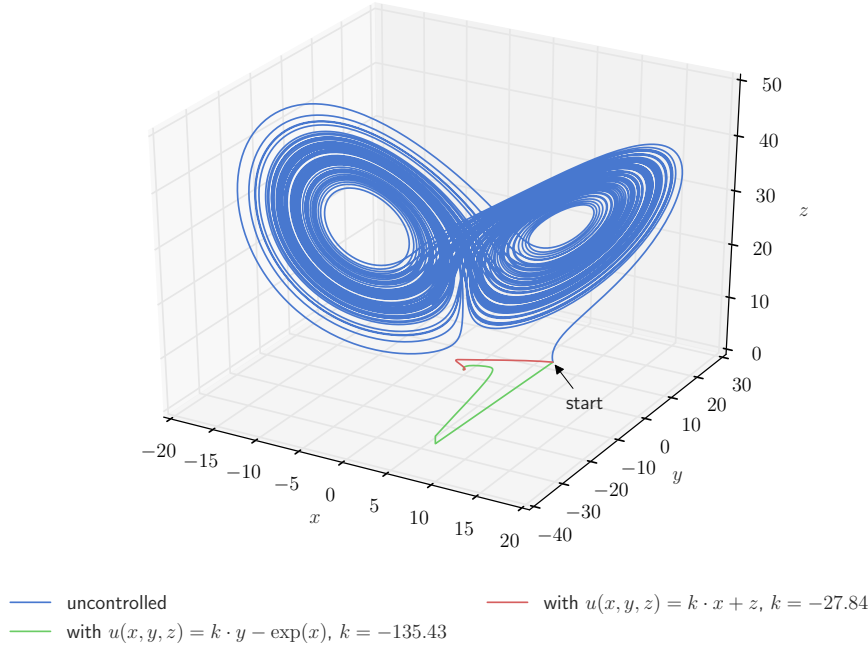


Fig. 10 Phase portrait of the forced Lorenz system with control exerted in \dot{y} . (Green and red: The system trajectories when controlled by two particular Pareto-front solutions. Blue: the uncontrolled chaotic system.)

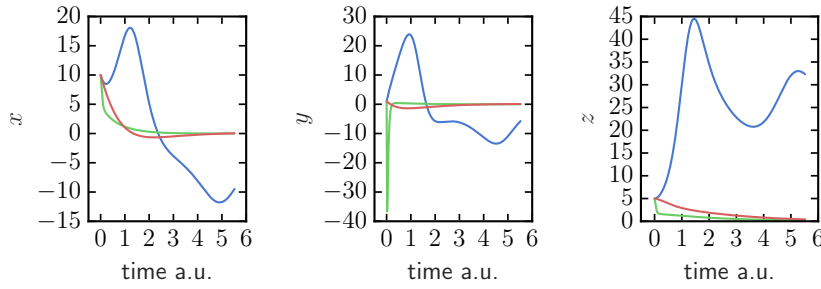


Fig. 11 Detailed view of the single trajectories in x , y , and z dimension. (blue: uncontrolled; green: $u(x, y, z) = -\exp(x) + k \cdot y$, $k = -135.43$; red: $u(x, y, z) = k \cdot x + z$, $k = -27.84$.)

by the indirect influence of z on the other state variables, hence, it is not surprising that the control laws here are more involved than in the previous case. Also, they generally do not perform well in the control of z , which is expressed by the relatively high values in RMSE_z . This is confirmed by the phase portrait of the solution $u(x, y, z) = -(k \cdot (-y) + x \cdot z + y + z)$ shown in

Figure 12: While going straight to the origin in the xy -plane there are strong oscillations of the trajectory along the z -axis.

The dynamics caused by the actuation, e.g. for the best control law found, can be explained qualitatively: there is a strong damping in all variables but y . This reflects the tendency to suppress z -oscillations and, at the same time, to add damping in y through the xz term: if y grows, the z contribution to damping on the right hand side of the Lorenz equations (7) grows and, in turn, damps y . This is, however, only possible to some extent, hence, the oscillations observed in Figure 12.

Table 6 Control of the Lorenz system in z : selected Pareto-front solutions.

RMSE _{x}	RMSE _{y}	RMSE _{z}	length	expression	constants
0.289289	0.139652	26.994070	13	$-(k \cdot (-y) + x \cdot z + y + z)$	$k = 793.129676$
0.327926	0.267043	27.070289	8	$\exp(-k + y \cdot \sin(y))$	$k = -4.254574$
0.431993	0.508829	32.116326	7	$(k + x) \cdot (y + z)$	$k = 2.638069$
0.471535	0.525010	26.986321	5	$k + x \cdot z$	$k = 67.137183$
0.637056	0.605686	26.895493	7	$\exp(k + y \cdot \sin(y))$	$k = 3.964478$
0.677204	0.703577	27.019308	4	$y + \exp(k)$	$k = 4.276256$
0.930668	0.952734	26.895126	5	$x + \exp(\exp(k))$	$k = 1.448198$
1.764030	1.860288	26.766383	6	$(k + x) \cdot \exp(y)$	$k = 21.783557$

4 From Coupled Oscillators to Networks

In this section we extend the above study and demonstrate the application of GP-based control to networks of phased oscillators. As discussed in the Introduction, such networks provide a powerful medium to modeling highly non-linear complex systems, including the human brain. Specifically, four cases are considered in this study in order of growing complexity. These include a system of two coupled oscillators; a 1D network of oscillators organized on a ring; a 2D network of oscillators organized on a closed torus; and a hierarchical network which has been proposed as a simplified model for the human brain [42].

The aim, for all systems under consideration, is to control the synchronization behavior of the coupled oscillators. This can be done in two ways: starting from a synchronization regime and forcing the system into de-synchronization or vice versa, i.e., starting from a de-synchronized regime and forcing the system into synchronization. Both control goals are evaluated for each of the four systems described above.

The synchronization of dynamical systems is a well-known phenomenon exhibited by diverse ensembles of oscillators and oscillatory media [6]. Here we will focus on a simple, but popular representative, the van der Pol oscillator; also used as a simple model for neurons. The van der Pol oscillator shows a

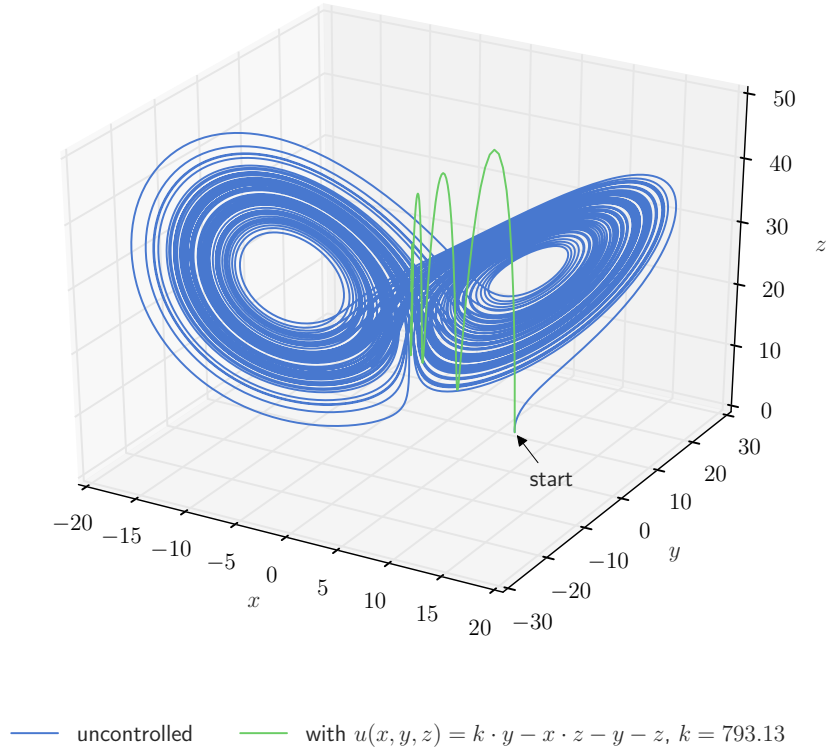


Fig. 12 Control of the Lorentz system in \dot{z} .

nonlinear damping behavior governed by the following second-order differential equation:

$$\ddot{x} = -\omega^2 x + \alpha \dot{x} (1 - \beta x^2) =: f_{\text{vdP}}(x, \dot{x}), \quad (8)$$

where x is the dynamical variable and $\omega, \alpha, \beta > 0$ are model parameters. The parameter ω is the characteristic frequency of the self-sustained oscillations, that is, the frequency at which the system tends to oscillate in the absence of any driving or damping force. The parameter α controls the non-linearity of the system. When $\alpha = 0$, Equation (8) becomes the harmonic oscillator. The damping parameter β controls the dilation of the trajectory in the phase space. See Figure 13.

A wide variety of coupling mechanisms exist. For hierarchical networks the investigation can be restricted to linearly coupled van der Pol oscillators in x and \dot{x} , which can be described by a global coupling constant and two coupling matrices. An uncontrolled system of N coupled van der Pol oscillators can be

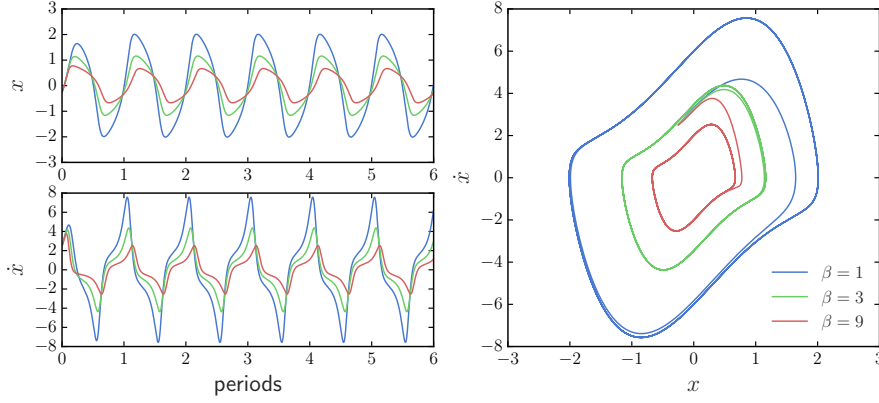


Fig. 13 The single van der Pol oscillator. (With parameters $\omega = e^2$, $\alpha = 3$, and initial conditions $x(0) = -0.25$, $\dot{x}(0) = 2.5$.)

stated as follows:

$$\ddot{x}_i = f_{\text{vdP}}(x_i, \dot{x}_i) + c_1 \sum_{j=0}^{N-1} \kappa_{ij} x_j + c_2 \sum_{j=0}^{N-1} \varepsilon_{ij} \dot{x}_j \quad (i = 0, \dots, N-1) \quad (9)$$

with initial conditions

$$x_i(t_0) = x_{i,0}, \quad \dot{x}_i(t_0) = \dot{x}_{i,0},$$

where $c_{1,2}$ are the global coupling constants and (κ_{ij}) and (ε_{ij}) are the respective coupling matrices. This allows for several types of coupling such as direct, diffusive, and global coupling, or any other kind of network-like coupling. In the following experiments, we will use diffusive coupling in \dot{x}_i for the four network topologies mentioned above. For GP, we use the same setup described above (Section 2.2.3).

4.1 Two Coupled Oscillators

The simplest system showing synchronization is a system of two diffusely coupled van der Pol oscillators:

$$\begin{aligned} \ddot{x}_0 &= -\omega_0^2 x_0 + \alpha \dot{x}_0 (1 - \beta x_0^2) + c(\dot{x}_1 - \dot{x}_0), \\ \ddot{x}_1 &= -\omega_1^2 x_1 + \alpha \dot{x}_1 (1 - \beta x_1^2) + c(\dot{x}_0 - \dot{x}_1). \end{aligned} \quad (10)$$

The coupling is restricted to \dot{x}_i , in which case the coupling constants from (9) are set to $c_1 = 0$, $c_2 = c$, and the remaining coupling matrix reads as follows: $(\varepsilon_{ij}) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$.

For the uncoupled oscillators there are some parameter combinations (α, β) for which there exist stable limit cycles with characteristic frequencies $\omega_{0,1}$.

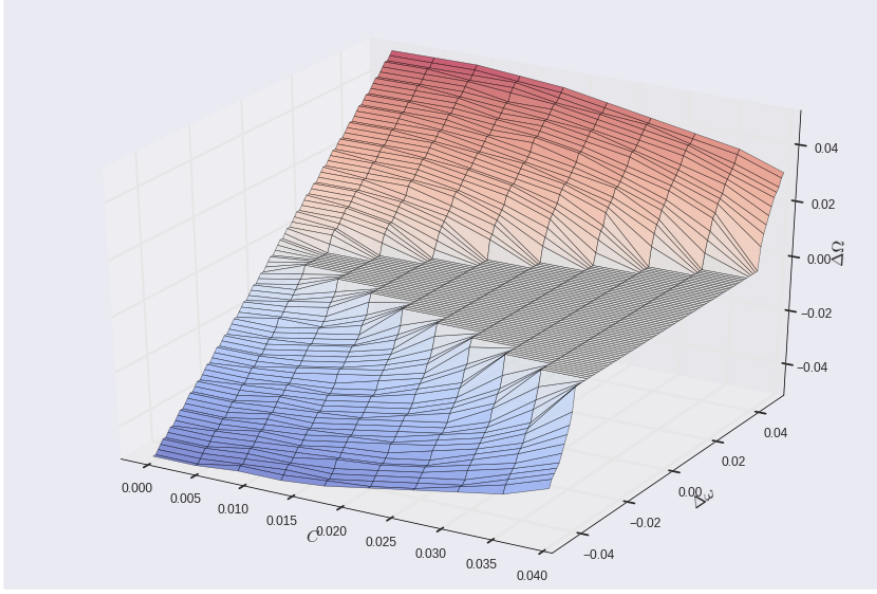


Fig. 14 Synchronization plot of two coupled van der Pol oscillators with varying coupling strength c . The horizontal V-shaped plateau is referred to as the Arnold tongue, it represents regions of synchronization. (The parameter set and initial conditions used are stated in the left part of Table 7.)

If the two oscillators are coupled by a given coupling constant $c \neq 0$, as in (10), a range of frequencies with $\omega_0 \neq \omega_1$ emerge, where both oscillators effectively oscillate in a common mode. This range of frequencies is called the synchronization region. With the variation in the coupling constant this region changes in width.

To illustrate this phenomenon, consider the particular parameter set $\alpha = 0.1$, $\beta = 1$, with fixed $\omega_0 = 1.386$ and varying ω_1 in the range $[\omega_0 - 0.06, \omega_0 + 0.06]$. By plotting the observed frequency difference¹ $\Delta\Omega$, exhibited by the two oscillators, against the difference in their characteristic frequencies, $\Delta\omega := \omega_1 - \omega_0$, we can visualize the synchronization behavior of the system for a given coupling constant. See Figure 14. Regions of synchronization show up as horizontal segments at $\Delta\Omega = 0$ (also, note the symmetry about $\Delta\omega = 0$). If this is done for several values of c in the range $[0, 0.4]$, we can trace out the regions of synchronization. The result is a typical V-shaped plateau, the Arnold tongue.

Figure 14 shows the choice of appropriate parameters ω_1 and c to setup the system in different regimes for the purpose of control. The same approach is taken for all the experiments presented in this section.

¹ The actual frequency Ω of an oscillator can be determined numerically by either taking the Fourier transform of the trajectory $x(t)$, or by counting the zero-crossings of $x(t) - \langle x(t) \rangle$.

Similar to the example systems in Section 3 we add the control function u to the equations (10) of the uncontrolled system, yielding the following formulation:

$$\begin{aligned}\ddot{x}_0 &= -\omega_0^2 x_0 + \alpha \dot{x}_0 (1 - \beta x_0^2) + c(\dot{x}_1 - \dot{x}_0) + u(\dot{\mathbf{x}}), \\ \ddot{x}_1 &= -\omega_1^2 x_1 + \alpha \dot{x}_1 (1 - \beta x_1^2) + c(\dot{x}_0 - \dot{x}_1) + u(\dot{\mathbf{x}}).\end{aligned}\quad (11)$$

Here, u is added as a global actuator term with equal influence on both oscillators; u may depend on \dot{x}_0 and \dot{x}_1 , summarized in vector notation as $\dot{\mathbf{x}} = (\dot{x}_0, \dot{x}_1)$.

4.1.1 Forced Synchronization

The system setup for forced synchronization of the two coupled van der Pol oscillators is presented in Table 7. The parameters ω_1 and c are chosen according to Figure 14, such that the uncontrolled system follows a de-synchronization regime at a distance, $\Delta\omega$, approximately half the plateau from the closest synchronization point. The initial conditions are the same for both oscillators.

The degree of de-synchronization is encompassed by the following cost functional:

$$\Gamma_1 := |\Omega_0 - \Omega_1|. \quad (12)$$

It measures the difference in observed frequencies exhibited by the two oscillators, with smaller differences reducing the cost on this objective.

As stated in the previous section, the actual frequencies Ω_0 and Ω_1 are numerically determined by counting zero crossings of the trajectory $x - \langle x \rangle$. This requires a careful choice of the time range $[t_0, t_n]$ of the observations, since, the number of periods N_P fitting into this interval determines an upper bound in absolute accuracy ($\sim \frac{1}{2N_P}$) of measuring Ω_0, Ω_1 . Here, $N_P = 2000$ is chosen to yield an absolute accuracy well below 10^{-3} in the frequency range of interest.

Table 7 Two Coupled Oscillators: system setup for forced synchronization.

dynamic system		GP	
ω_0	$\ln(4)$	cost functionals	$ \Omega_0 - \Omega_1 $
ω_1	$\ln(4) + 0.04$		$\text{length}(u)$
α, β, c	0.1, 1, 0.022	argument set	$\{\dot{x}_0, \dot{x}_1\}$
$\mathbf{x}(t_0)$	(1, 1)	constant set	$\{k\}$
$\dot{\mathbf{x}}(t_0)$	(0, 0)	seed	3464542173339676227
t_0, t_n	0, $2000 \frac{2\pi}{\omega_0}$		
n	40000		

Results from the GP run are presented in Table 8. The algorithm stopped after one generation, providing six simple results optimally satisfying Γ_1 . Since

the equations (11) are symmetric in x_0 and x_1 , unsurprisingly so, are the resulting control laws.

To demonstrate the control effect Figure 15 shows the Kuramoto order parameter, r , representing phase-coherence, plotted over time [43,44]. The order parameter is defined by

$$r = \left| \frac{1}{N} \sum_{j=0}^{N-1} e^{i\varphi_j} \right|,$$

with φ_j the continuous phase² of the j -th oscillator. The controlled system completely synchronizes ($r \approx 1$) after a short initial period of de-synchronization, while the uncontrolled system exhibits a permanent phase shift resulting in an oscillating graph.

Table 8 Two Coupled Oscillators: Pareto-front solutions for forced synchronization.

$ \Omega_0 - \Omega_1 $	length	expression
0.0	2	$\cos(\dot{x}_1)$
0.0	2	$\cos(\dot{x}_0)$
0.0	2	$-\dot{x}_0$
0.0	2	$\sin(\dot{x}_1)$
0.0	2	$-\dot{x}_1$
0.0	2	$\sin(\dot{x}_0)$

One recognizes that the algorithm favors the least complex solution, using an asymmetric term, either damping in x_1 or x_0 . We can analyze qualitatively these solutions, $u(\dot{\mathbf{x}}) = -\dot{x}_0$ and $u(\dot{\mathbf{x}}) = -\dot{x}_1$. Let us take arbitrarily the x_0 term: plugging in $u(\dot{\mathbf{x}}) = -\dot{x}_0$ into the first oscillator equation from (11) we obtain

$$\begin{aligned} \ddot{x}_0 &= -\omega_0^2 x_0 + \alpha \dot{x}_0 (1 - \beta x_0^2) + c(\dot{x}_1 - \dot{x}_0) - \dot{x}_0 \\ &= -\omega_0^2 x_0 + (\alpha - c - 1)\dot{x}_0 - \alpha\beta\dot{x}_0 x_0^2 + c\dot{x}_1 \\ &= -\omega_0^2 x_0 + \tilde{\alpha}_1 \dot{x}_0 - \alpha\beta\dot{x}_0 x_0^2 + c\dot{x}_1, \end{aligned}$$

with $\tilde{\alpha}_1 = \alpha - c - 1$. Doing the same for the second oscillator equation

$$\begin{aligned} \ddot{x}_1 &= -\omega_1^2 x_1 + \alpha \dot{x}_1 (1 - \beta x_1^2) + c(\dot{x}_0 - \dot{x}_1) - \dot{x}_1 \\ &= -\omega_1^2 x_1 + (c - 1)\dot{x}_0 - \alpha\beta\dot{x}_1 x_1^2 + (\alpha - c)\dot{x}_1 \\ &= -\omega_1^2 x_1 + \tilde{\alpha}_2 \dot{x}_0 - \alpha\beta\dot{x}_1 x_1^2 + \tilde{c}\dot{x}_1, \end{aligned}$$

with $\tilde{\alpha}_2 = c - 1$ and $\tilde{c} = \alpha - c$, one gets a similar solution in terms of the dominating driving component \dot{x}_0 . Since $\tilde{\alpha}_1 \approx \tilde{\alpha}_2$ and $\tilde{\alpha}_1 \gg \tilde{c}, c$, the oscillators behave almost identically and, thus, are synchronized. Analogous observations apply when analyzing the second control law, $u(\dot{\mathbf{x}}) = -\dot{x}_1$.

² The continuous phase is computed from the analytical signal of the trajectory x using the Hilbert transform. For details see [45].

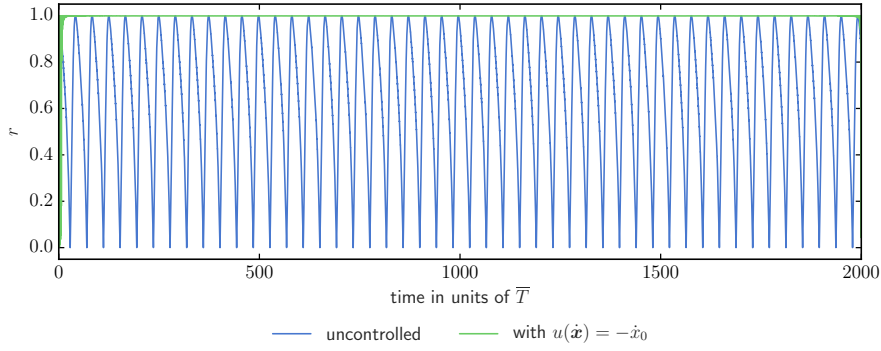


Fig. 15 Two Coupled Oscillators: Kuramoto order parameter, r , for forced synchronization. Green: the controlled, and blue: the uncontrolled system.

4.1.2 Forced De-Synchronization

The system setup for forced de-synchronization is given in Table 7. The parameters ω_1 and c are, again, chosen according to Figure 14, this time, such that the uncontrolled system follows a synchronization regime well inside the plateau. The measure for the degree of synchronization is reciprocal to the previous case

$$\Gamma_1 := \exp(-|\Omega_0 - \Omega_1|). \quad (13)$$

All the remaining parameters are the same as for forced synchronization.

Table 9 Two Coupled Oscillators: System setup for forced de-synchronization.

dynamic system		GP	
ω_0	$\ln(4)$	cost functionals	$\exp(- \Omega_0 - \Omega_1)$
ω_1	$\ln(4) + 0.015$		$\text{length}(u)$
α, β, c	0.1, 1, 0.022	argument set	$\{\dot{x}_0, \dot{x}_1\}$
$\mathbf{x}(t_0)$	(1, 1)	constant set	$\{k\}$
$\dot{\mathbf{x}}(t_0)$	(0, 0)	seed	2590675513212712687
t_0, t_n	0, $2000 \frac{2\pi}{\omega_0}$		
n	40000		

Results from the GP run are shown in Table 10. Two aspects of the results indicate that de-synchronizing the pair of oscillators is a more demanding task: First, the GP algorithm comes up with increasingly long expressions to achieve improvements in Γ_1 ; second, constant optimization seems to fail in all cases where a constant is present (this is expressed by a value $k = 1$, which corresponds to the initial guess of the optimization procedure). Still, the oscillating Kuramoto parameter, r , of the controlled system in Figure 16 shows,

that the best solution with respect to I_1 performs well in de-synchronizing the oscillators.

The control law $u(\dot{\mathbf{x}}) = -\dot{x}_0 \cdot \exp(\exp(k) + \cos(k)) = -\tilde{k}\dot{x}_0$, with $\tilde{k} \approx 26$, is almost the same as the solution analyzed in the previous subsection, but with a different coefficient. This is at first sight counterintuitive, but can be explained roughly by the non uniqueness we provoke with our cost function: to force synchronization we require only that the phase difference is small (close to zero). This is achieved by the added damping term. The very strong damping brings the two oscillators basically to zero so fast that the mutual coupling does not play a role and the phase difference is free. To bring the oscillators from desynchronization to synchronization is achieved by a different mechanism; the damping is moderate such that excess energy is dissipated and the oscillators are in the right regime to synchronize. The detailed analysis of the dynamics is subject of ongoing work, where we analyze the bifurcations occurring using AUTO. Preliminary results affirm that the interpretation given here is correct.

One would expect the GP algorithm to directly generate the simpler – thus better adapted – solution $u(\dot{\mathbf{x}}) = -k\dot{x}_0$, with $k = 26$. It is not entirely clear, why this is not the case here. One possible explanation might lie with the constant optimization algorithm: on failure, the least squares algorithm returns the result of the last internal iteration. This return value might be an entirely inadequate value for k , which, in turn, could lead to an exploding cost-index I_1 when integrating the dynamic system (11), hence, disqualifying the corresponding solution.

Table 10 Two Coupled Oscillators: Pareto-front solutions for forced de-synchronization.

$\exp(- \Omega_0 - \Omega_1)$	length	expression	constant
0.248	9	$-\dot{x}_0 \cdot \exp(\exp(k) + \cos(k))$	$k = 1$
0.258	7	$\cos(\exp(\dot{x}_1 + \cos(\cos(\dot{x}_0))))$	
0.875	4	$\cos(\exp(\exp(\dot{x}_0)))$	
0.912	3	$\sin(\exp(\dot{x}_0))$	
0.999	2	$\exp(k)$	$k = 1$
1.000	1	\dot{x}_1	
1.000	1	\dot{x}_0	
1.000	1	k	$k = 1$
0.247672	13	$-\dot{x}_0 \cdot \exp(-\dot{x}_0) \cdot \exp(\exp(k)) - \exp(\dot{x}_0)$	$k = 1.000000$
0.386623	12	$-\exp(\dot{x}_0) - \cos(\exp(\exp(k)) \cdot \exp(\sin(\dot{x}_0)))$	$k = 1.000000$
0.398873	11	$\sin(\exp(\exp(\dot{x}_0) + \cos(k)) \cdot \exp(\sin(\dot{x}_0)))$	$k = 1.000000$
0.606677	8	$\sin(\exp(\exp(k)) \cdot \exp(\sin(\dot{x}_0)))$	$k = 1.000000$
0.648420	7	$\sin(\exp(\exp(\dot{x}_0) + \cos(k)))$	$k = 1.000000$
0.806083	6	$-\exp(\dot{x}_0) - \cos(k)$	$k = 1.000000$
0.911933	3	$\sin(\exp(\dot{x}_0))$	$k = 1.000000$
0.998615	2	$\exp(k)$	$k = 1.000000$
1.000000	1	\dot{x}_1	$k = 1.000000$
1.000000	1	\dot{x}_0	$k = 1.000000$
1.000000	1	k	$k = 1.000000$

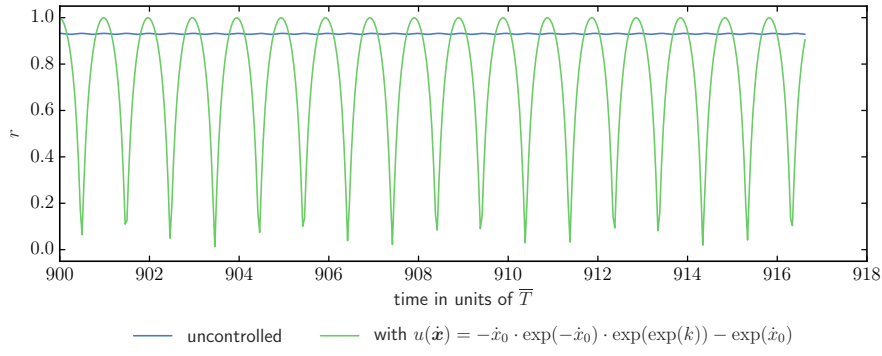


Fig. 16 Two Coupled Oscillators: Kuramoto order parameter, r , for forced desynchronization. Green: the controlled, and blue: the uncontrolled system. The horizontal axis is scaled to a limited time window in order to make the oscillations visible.

4.2 One dimensional networks of oscillators

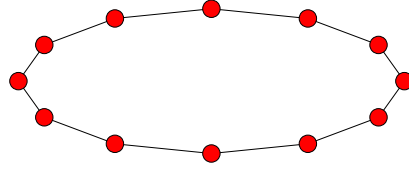


Fig. 17 Ring topology.

Now, the previous system is extended by using a ring-like topology of twelve van der Pol oscillators [22]. The corresponding equations of the controlled dynamic system read

$$\ddot{x}_i = -\omega_i^2 x_i + \alpha \dot{x}_i (1 - \beta x_i^2) + c \sum_{j=0}^{N-1} \varepsilon_{ij} \dot{x}_j + u(\dot{\mathbf{x}}) \quad (i = 0, \dots, N-1), \quad (14)$$

with $N = 12$ and the 12×12 coupling matrix defined as

$$(\varepsilon_{i,j}) = \begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & & \ddots & & \\ & & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}.$$

The coupling matrix reproduces the topology shown in Figure 17, where each oscillator is diffusively coupled in \dot{x}_i to its two neighbors \dot{x}_{i-1} , \dot{x}_{i+1} ($i \in \mathbb{Z}/N\mathbb{Z}$). The degree of synchronization is governed by the coupling strength c and the distribution of characteristic frequencies ω_i .

4.2.1 Forced Synchronization

The system parameters for forced synchronization are given in Table 11. They are chosen first roughly according to the synchronization regime determined in Figure 14, and it is carefully checked that the starting system state really lies in the desynchronized regime. The $N = 12$ characteristic frequencies of the single oscillators are equally spaced in the interval $[\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega]$ (denoted $\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 12)$ in Table 11).

The GP setup is similar to that for the paired van der Pol oscillators in Section 4.1. Their are two main differences concerning the increase in the number of oscillators: First, a reformulation of the cost functional for $N > 2$. For that purpose the modulus in (12) is replaced by the standard deviation of the observed oscillator frequencies

$$\Gamma_1 := \text{std}(\boldsymbol{\Omega}) = \frac{1}{N} \sum_{i=0}^{N-1} (\Omega_i - \langle \boldsymbol{\Omega} \rangle)^2, \quad (15)$$

with $N = 12$, in this case. Second, an extension of the argument set to incorporate \dot{x}_i ($i = 1, \dots, N$), that is, the control function u is given access to ideal measurements from all twelve state variables.

Table 11 Oscillators on a Ring: System setup for forced synchronization.

dynamic system		GP	
α, β, c	0.1, 1, $4.11 \cdot 10^{-2}$	cost functionals	$\text{std}(\boldsymbol{\Omega})$
$\bar{\omega}, \Delta\omega$	$\ln(4)$, $2 \cdot 10^{-2}$		$\text{length}(u)$
ω_i	$\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 12)$	argument set	$\{\dot{x}_i\}_{i=0, \dots, 11}$
$\mathbf{x}_i(t_0)$	1 ($i = 0, \dots, 11$)	constant set	$\{k\}$
$\dot{\mathbf{x}}_i(t_0)$	0 ($i = 0, \dots, 11$)	seed	7612506836576153849
t_0, t_n	0, $2000 \frac{2\pi}{\bar{\omega}}$		
n	40000		

The results from the GP run are listed in Table 12. The GP algorithm stopped after the first iteration, i.e. all solutions optimally satisfy the synchronization objective within precision of measurement offhand. The control laws found are all sinusoidal in nature, only differing in the particular argument \dot{x}_i . Because of the ring topology, where all nodes are of equal status, this is not surprising and might be expected from the optimization algorithm. Clearly, the MLC approach works well in this case. A graphical illustration of control with the particular solution $u(\dot{\mathbf{x}}) = \cos(\dot{x}_6)$ (first row of Table 12) is given in

Figure 18. It shows a comparison of the uncontrolled and controlled system. In the respective figures the amplitudes of the single oscillators are plotted side-by-side. The uncontrolled system exhibits irregular phase shifts and fluctuating amplitudes. The controlled system, however, seems to be perfectly aligned in phase. The amplitudes are slightly increased overall and display a decreasing trend from low to high oscillator index.

A more detailed view is given by the Kuramoto parameter, shown in Figure 19. It reveals that perfect synchronization is not achieved (small oscillations in the graph), which may be explained by the solutions found by the GP algorithm: instead of the single driving by \dot{x}_i a sinusoidal control law is found, which results in suboptimal synchronization.

Table 12 Oscillators on a Ring: Pareto-front solutions for forced synchronization.

std(Ω)	length	expression
0.0	2	$\cos(\dot{x}_6)$
0.0	2	$\sin(\dot{x}_5)$
0.0	2	$\sin(\dot{x}_9)$
0.0	2	$\sin(\dot{x}_1)$
0.0	2	$\sin(\dot{x}_6)$
0.0	2	$\sin(\dot{x}_{11})$
0.0	2	$\sin(\dot{x}_3)$

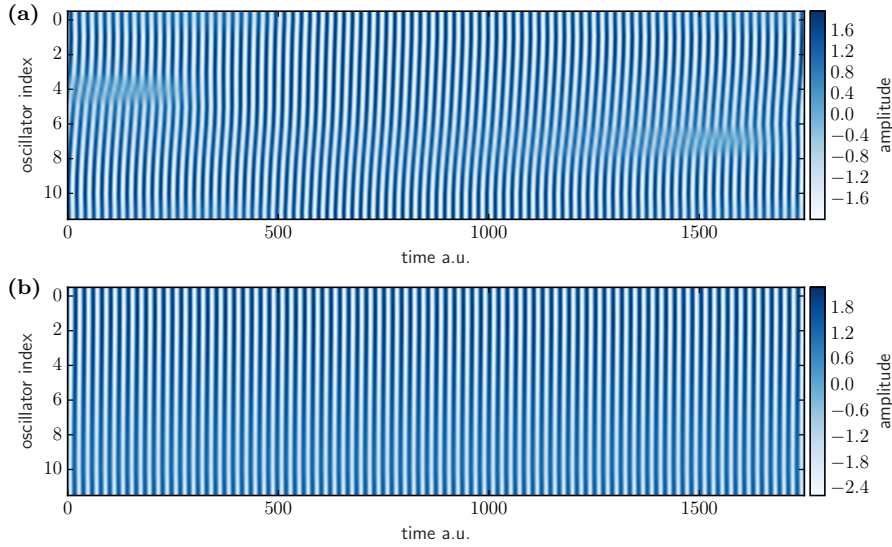


Fig. 18 Oscillators on a Ring. **a:** uncontrolled system; **b:** Pareto-front solution, $u(\dot{x}) = \cos(\dot{x}_6)$, for forced synchronization.

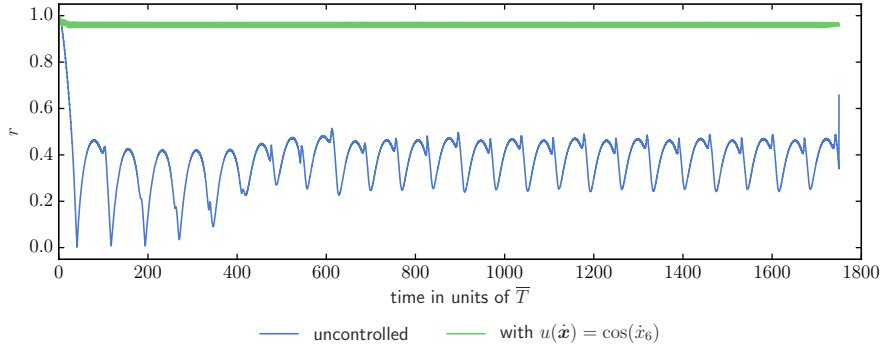


Fig. 19 Oscillators on a Ring. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to F_1 (green).

4.2.2 Forced De-Synchronization

The system parameters for forced de-synchronization are given in Table 13. Compared to the synchronization case, the only difference lies in the two parameters c and $\Delta\omega$, which are chosen, such that the uncontrolled system resides in a synchronization regime. The remaining parameters are the same as in the case of forced synchronization.

Again, the cost functional from the paired van der Pol oscillators (13) serves as template for the general case of N oscillators

$$F_1 := \exp(-\text{std}(\mathbf{\Omega})), \quad (16)$$

with $N = 12$ here.

Table 13 Oscillators on a Ring: System setup for forced de-synchronization.

dynamic system		GP	
α, β, c	0.1, 1, $3.33 \cdot 10^{-2}$	cost functionals	$\exp(-\text{std}(\mathbf{\Omega}))$
$\bar{\omega}, \Delta\omega$	$\ln(4), 2 \cdot 10^{-3}$		$\text{length}(u)$
ω_i	$\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 12)$	argument set	$\{\dot{x}_i\}_{i=0,\dots,11}$
$\mathbf{x}_i(t_0)$	1 ($i = 0, \dots, 11$)	constant set	$\{k\}$
$\dot{\mathbf{x}}_i(t_0)$	0 ($i = 0, \dots, 11$)	seed	5212795243969009234
t_0, t_n	0, $2000 \frac{2\pi}{\bar{\omega}}$		
n	40000		

The results of the GP run for forced de-synchronization are listed in Table 14. Of course, according to the above, in the context of suppression of collective oscillations, the desynchronization is very important for an application. We recognize that there exist complex terms containing two dissipative terms (first and third solution), and most of the solutions which depend only

on one oscillators derivative. Since we did not require any symmetry in our const function, here again it seems the algorithm chooses the “one takes it all” strategy. The simple solutions with only additive damping are basically synchronizaed, whereas the complex ones show significant desynchronization. The most important term is probably the $\cos(\sin(\dot{x}_2))$ one, it applies a variation in damping such that synchronization is destroyed by this variation.

Table 14 Oscillators on a Ring: Pareto-front solutions for forced de-synchronization.

$\exp(-\text{std}(\Omega))$	length	expression	constant
0.473	9	$-\exp(\exp(\dot{x}_5)) + \exp(\cos(\sin(\dot{x}_2)))$	$k = 1$
0.683	7	$-k - \dot{x}_{11} - \sin(k)$	
0.794	6	$-\dot{x}_{11} + \dot{x}_{11} \cdot \dot{x}_6$	
0.958	4	$\cos(\exp(\exp(\dot{x}_1)))$	
0.978	3	$\sin(\exp(\dot{x}_1))$	
0.997	2	$-\dot{x}_5$	
1.000	1	\dot{x}_1	
1.000	1	\dot{x}_8	
1.000	1	\dot{x}_{10}	
1.000	1	\dot{x}_4	

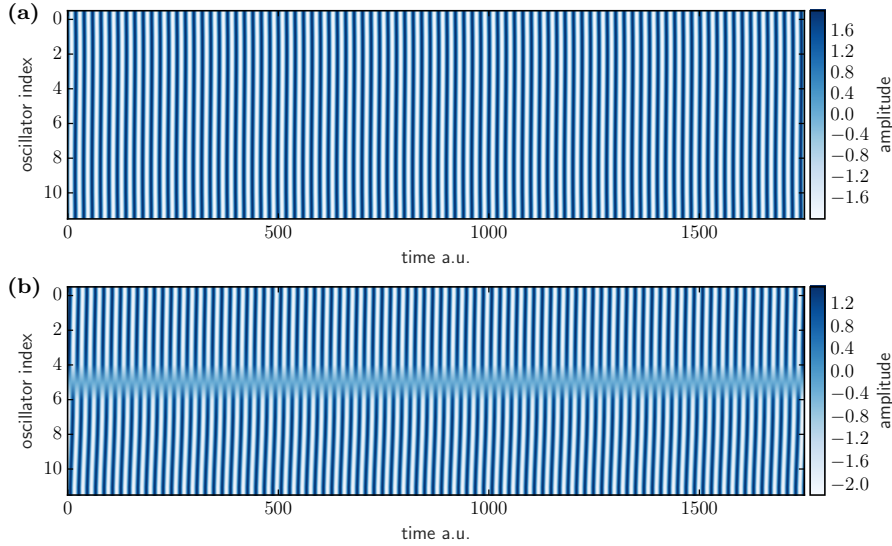


Fig. 20 Oscillators on a Ring. **a:** uncontrolled system; **b:** Pareto-front solution, $u(\dot{\mathbf{x}}) = -\exp(\exp(\dot{x}_5)) + \exp(\cos(\sin(\dot{x}_2)))$, for forced de-synchronization.

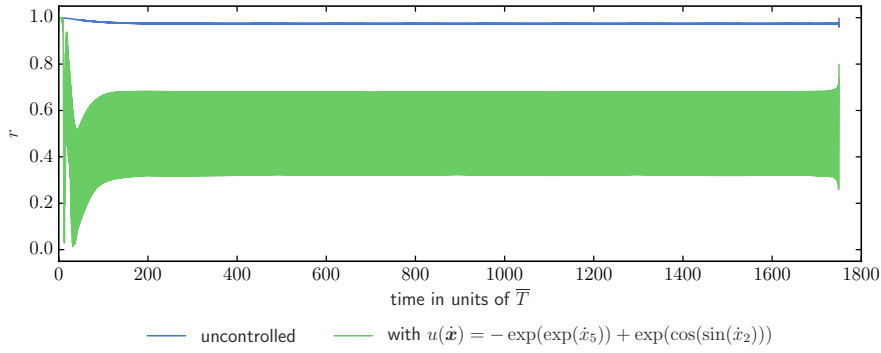


Fig. 21 Oscillators on a Ring. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to F_1 (green).

4.3 Two dimensional networks

Next, the ring system from the previous section is extended to a 2D torus-like topology of 12×12 van der Pol oscillators. The dynamic system can be again described by (14), but with $N = 144$ and a different coupling matrix in place. Since a formal statement of the 144×144 coupling matrix is laborious and does not add to a clear understanding, it is omitted here (however it is easily implemented numerically). The matrix diffusively couples every oscillator with four of its nearest neighbors on the torus. See Figure 22 for a depiction of the topology. The nodes follow a linear row-by-row ordering, that is, nodes from the first row are labeled from left to right $i = 0, \dots, 11$, and nodes from the second row $i = 12, \dots, 23$, and so on, till the bottom right node with index $i = 143$.

The goal is again to force the system either into synchronization or de-synchronization. In contrast to the previous topologies the argument set does not contain the \dot{x}_i from all oscillators. The potential control functions u have to resort to the \dot{x}_i from the first twelve oscillators ($i = 0, \dots, 11$) as input. This is an arbitrary choice, however, it comes closer to real world experiments where only a subset of relevant state variables of a system might be accessible to the experimenter.

4.3.1 Forced Synchronization

The system setup for forced synchronization follows along the same lines as for the ring topology in Section 4.2.1: The chosen parameter set (in particular $\Delta\omega$, c) puts the uncontrolled system in a de-synchronization regime. The cost functional F_1 rates synchronicity by the standard deviation in oscillator frequencies. Constant optimization is performed on a single constant.

Table 16 lists the results from the GP run. The algorithm stopped iteration after the first generation. As in the case of forced synchronization for the ring

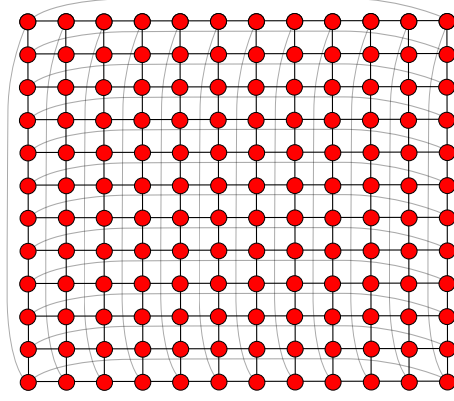


Fig. 22 Torus topology: a 12×12 grid with periodic boundary conditions.

Table 15 Oscillators on a Torus: System setup for forced synchronization.

dynamic system		GP	
α, β, c	0.1, 1, 0.218	cost functionals	$\text{std}(\mathbf{I})$
$\bar{\omega}, \Delta\omega$	$\ln(4), 5.5 \cdot 10^{-2}$		$\text{length}(u)$
ω_i	$\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 144)$	argument set	$\{\dot{x}_i\}_{i=0, \dots, 11}$
$\mathbf{x}_i(t_0)$	1 ($i = 0, \dots, 143$)	constant set	$\{k\}$
$\dot{\mathbf{x}}_i(t_0)$	0 ($i = 0, \dots, 143$)	seed	6349713261226598530
t_0, t_n	0, $2000 \frac{2\pi}{\bar{\omega}}$		
n	40000		

topology (Section 4.2.1), all-out sinusoidal control laws are found. The symmetry of the torus topology makes all \dot{x}_i equally viable inputs, as is reflected by the solutions optimally satisfying I_1 regardless of the particular argument \dot{x}_i .

In Figure 23 we find by eye already a good phase alignment of the oscillators for the controlled system. Amplitudes are magnified ($\times 1.5$) and the phase is matched, when compared to the uncontrolled system. The Kuramoto order parameter in Figure 24 reveals that perfect synchronization is not achieved, as can be seen by the small oscillations exhibited in the graph of the controlled system.

All control laws found are of complexity 2 and are of the form $\sin(\dot{x}_i)$. This is a comparatively strong control, such that the one oscillator starts controlling the others which is in accordance with our cost function.

4.3.2 Forced De-Synchronization

For forced de-synchronization the interval of characteristic frequencies is narrowed down, compared to the previous case of synchronization, and the coupling constant c is decreased. This causes the system of uncontrolled oscillators to shift into a synchronization regime. All other parameters remain the same.

Table 16 Oscillators on a Torus: Pareto-front solutions for forced synchronization.

$\text{std}(\boldsymbol{\Omega})$	length	expression
0.0	2	$\sin(\dot{x}_1)$
0.0	2	$\sin(\dot{x}_6)$
0.0	2	$\sin(\dot{x}_7)$
0.0	2	$\sin(\dot{x}_0)$
0.0	2	$\sin(\dot{x}_2)$
0.0	2	$\sin(\dot{x}_{11})$

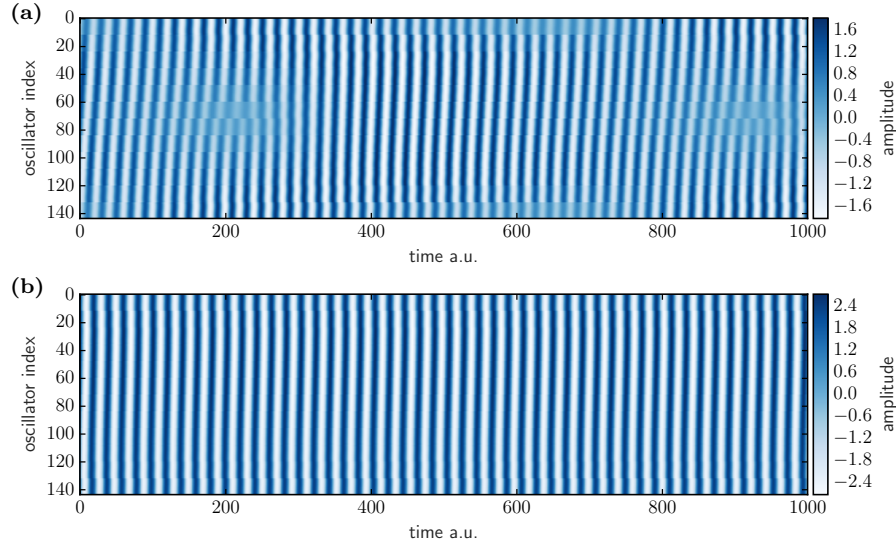
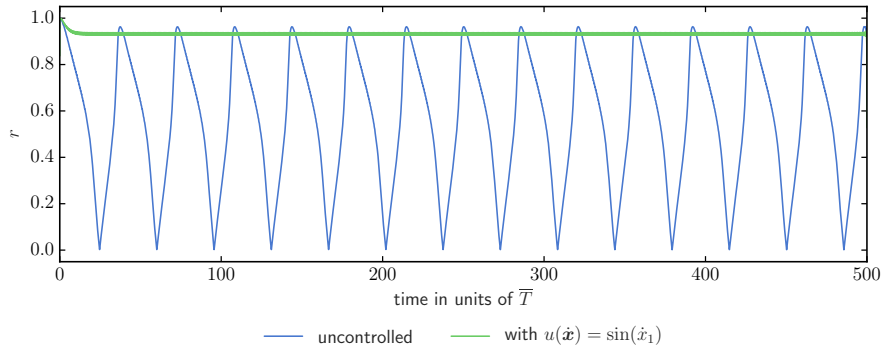
**Fig. 23** Oscillators on a Torus. **a**: uncontrolled system; **b**: Pareto-front solution, $u(\dot{\mathbf{x}}) = \sin(\dot{x}_1)$, for forced synchronization.**Fig. 24** Oscillators on a Torus. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to Γ_1 (green).

Table 17 Oscillators on a Torus: System setup for forced de-synchronization.

dynamic system		GP	
α, β, c	0.1, 1, 0.125	cost functionals	$\exp(-\text{std}(\boldsymbol{\Omega}))$
$\bar{\omega}, \Delta\omega$	$\ln(4), 1.5 \cdot 10^{-2}$		$\text{length}(u)$
ω_i	$\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 144)$	argument set	$\{\dot{x}_i\}_{i=0,\dots,11}$
$\mathbf{x}_i(t_0)$	1 ($i = 0, \dots, 143$)	constant set	$\{k\}$
$\dot{\mathbf{x}}_i(t_0)$	0 ($i = 0, \dots, 143$)	seed	5028849844344362202
t_0, t_n	0, $2000 \frac{2\pi}{\bar{\omega}}$		
n	40000		

By inspection of Table 17 we see that in comparison to the ring setup the torus needs more complex combinations of terms to desynchronize the dynamics. simple terms of low complexity are not sufficient, rather a -pretty unusual-combination of functions is found by our algorithm. The control term basically induces chaos in the solution by the highly nonlinear control. This indicates that we need a better formulated objective and probably a more extensive computational run to precise the objective and converge to a final solution, respectively.

Table 18 Oscillators on a Torus: Pareto-front solutions for forced de-synchronization.

$\exp(-\text{std}(\boldsymbol{\Omega}))$	length	expression	constant
0.763	20	$(\sin(\dot{x}_{10} \cdot \dot{x}_4) + \cos(\sin(\sin(\dot{x}_6) + \sin(\dot{x}_9)))) \cdot \exp(-\dot{x}_5 + \dot{x}_3 \cdot \dot{x}_8)$	
0.764	19	$(\sin(\dot{x}_{10} \cdot \dot{x}_4) + \cos(\sin(\dot{x}_8 + \sin(\dot{x}_6)))) \cdot \exp(-\dot{x}_5 + \dot{x}_3 \cdot \dot{x}_8)$	
0.783	15	$(\sin(\dot{x}_4 \cdot \dot{x}_8) + \cos(\dot{x}_8)) \cdot \exp(-\dot{x}_5 + \dot{x}_3 \cdot \dot{x}_8)$	
0.880	6	$-\dot{x}_8 - \sin(\exp(\dot{x}_6))$	
0.889	4	$-k - \dot{x}_8$	$k = 1$
0.891	3	$-\exp(\dot{x}_8)$	
0.999	2	$\exp(k)$	$k = 1$
1.000	1	\dot{x}_7	
1.000	1	\dot{x}_8	
1.000	1	k	$k = 1$
1.000	1	\dot{x}_3	

4.4 Hierarchical Network

In a last step the dynamic system is extended to a set of van der Pol oscillators connected in a scale-free network topology. A scale-free network is a hierarchical network whose degree distribution follows a power law, at least asymptotically. There exists a large variety of possible models for creating networks which are able to reproduce the unique properties of the scale-free

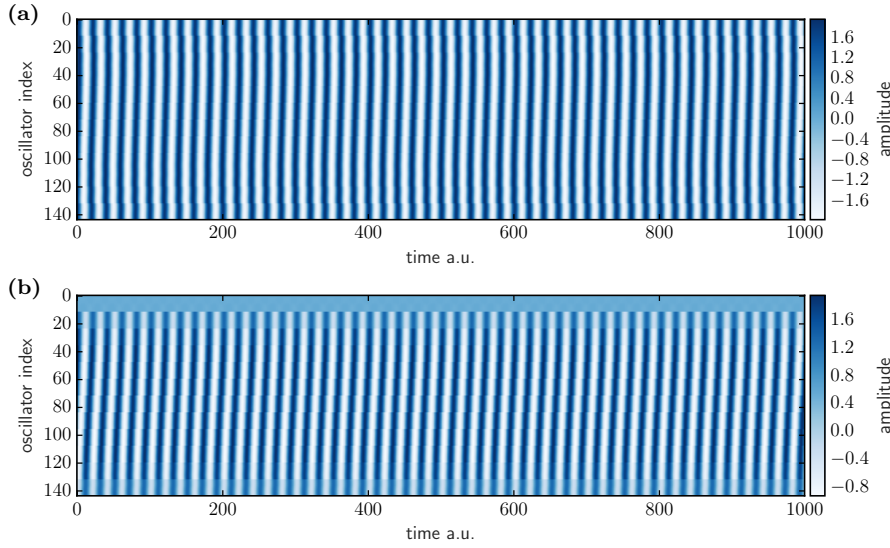


Fig. 25 Oscillators on a Torus. **a**: uncontrolled system; **b**: Pareto-front solution, $u(\dot{\mathbf{x}}) = (\sin(\dot{x}_{10} \cdot \dot{x}_4) + \cos(\sin(\sin(\dot{x}_6) + \sin(\dot{x}_9)))) \cdot \exp(\dot{x}_3 \cdot \dot{x}_8 - \dot{x}_5)$, for forced de-synchronization.

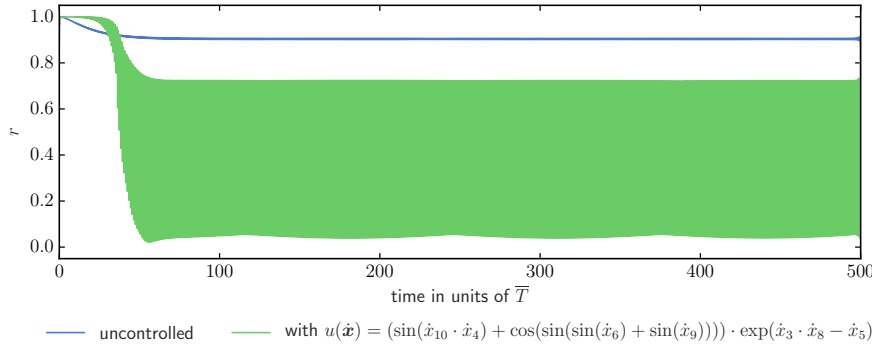


Fig. 26 Oscillators on a Torus. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to Γ_1 (green).

topology. One simple model, resorted to here, is the Dorogovtsev-Goltsev-Mendes model. It is used to produce a network of $N = 123$ nodes as depicted in Figure 27.

The van der Pol oscillators are indexed in descending order by their corresponding node degree. For example the three yellow nodes of degree 32 (highest) in Figure 27 are labeled $i = 0, 1, 2$, the light orange nodes of the next lowest degree 16, $i = 3, 4, 5$, and so on. The particular order of nodes of the same degree is not important due to the symmetry of the network.

“Sensors” are placed on the oscillators labeled $i = 0, \dots, 11$ measuring \dot{x}_i . This incorporates all nodes with a node degree of 32 or 16, and five nodes with

a node degree of 8. Hence, the control function u can potentially make use of measurements from nodes at central points of the topology. Control, on the other hand, is exerted indiscriminately on all nodes of the system

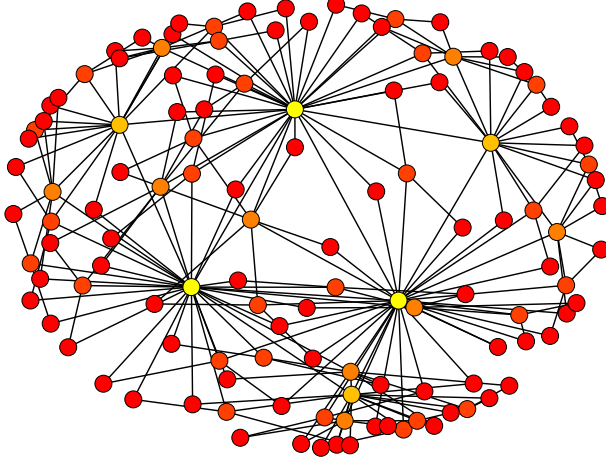


Fig. 27 Hierarchical Network: Dorogovtsev-Goltsev-Mendes topology of generation five. Starting out from two connected nodes at generation 0, one new node is added in between every existing pair of nodes per generation. Hence, the node degree at generation $n > 0$ ranges from $2^1, \dots, 2^n$. The degree distribution, that is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as $P(k) \sim k^{-2}$. (Nodes are color coded by node degree: yellow: 32, light orange: 16, orange: 8, dark orange: 4, red: 2.)

4.4.1 Forced Synchronization

Table 19 shows the setup. As in the related cases before, the particular parameter set chosen puts the uncontrolled system in a de-synchronization regime. Sensors are placed on the oscillators labeled $i = 0, \dots, 11$ measuring \dot{x}_i . This incorporates all nodes with a node degree of 32 or 16, and five nodes with a node degree of 8. Hence, the control function u can potentially make use of measurements from nodes at central points of the topology. Control, on the other hand, is exerted indiscriminately on all nodes of the system

$$\ddot{x}_i = f_{\text{vdP}}(x_i) + c \sum_{j=1}^N \varepsilon_{ij} \dot{x}_j + u(x_0, \dots, x_{11}) \quad (i = 0, \dots, N-1),$$

with $N = 123$. Constant optimization is performed on a single constant k .

Table 19 Oscillators in a hierarchical network: System setup for forced synchronization.

dynamic system		GP	
α, β, c	$0.1, 1, 5.6 \cdot 10^{-2}$	cost functionals	$\text{std}(\mathbf{J})$
$\bar{\omega}, \Delta\omega$	$\ln(4), 8 \cdot 10^{-2}$		$\text{length}(u)$
ω_i	$\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 123)$	argument set	$\{\dot{x}_i\}_{i=0,\dots,11}$
$\mathbf{x}_i(t_0)$	$1 \ (i = 0, \dots, 122)$	constant set	$\{k\}$
$\dot{\mathbf{x}}_i(t_0)$	$0 \ (i = 0, \dots, 122)$	seed	5925327490976859669
t_0, t_n	$0, 2000 \frac{2\pi}{\bar{\omega}}$		
n	40000		

The GP run stopped after one generation with a Pareto front consisting of a single optimal result. See Table 20. The control law found, is again, sinusoidal in nature. It uses the input from node $i = 7$, which is of node degree eight, i.e. on an intermediate level in the topology. Figure 28 shows, that the highly distorted phases from the uncontrolled system can be partially aligned. Frequencies are approximately matched and amplitudes are amplified by a factor $\times 1.5$ with respect to the uncontrolled system. A plot of the Kuramoto order parameter r in Figure 29 indicates, that there is a small variation among the phases in the controlled system, hence a perfect phase lock is not achieved.

Table 20 Oscillators in a hierarchical network: Pareto-front solutions for forced synchronization.

$\text{std}(\mathbf{J})$	length	expression
0.0	2	$\sin(\dot{x}_7)$

4.4.2 Forced De-Synchronization

For the case of forced de-synchronization the system parameters are set as in Table Table 21. The network structure stays as in the previous section.

The previous sections showed increasingly complex control from two oscillators, a ring of oscillators, to a torus. One might expect an even more complex control for a hierarchical network. Indeed, in Table Table 22 the best control laws are complicated combinations of sine, multiplication and exponential terms. As explained above, the lowest indices indicate highest node degree. We observe that our algorithm puts control on nodes with high degree - the hubs. This is perfectly logical: if the hubs are desynchronized, the whole system is desynchronized. It may still be that the hubs and their connected subnet are synchronized. This is not contained in our objective and again we find that our machine learns exactly the way it is told. If we want to control

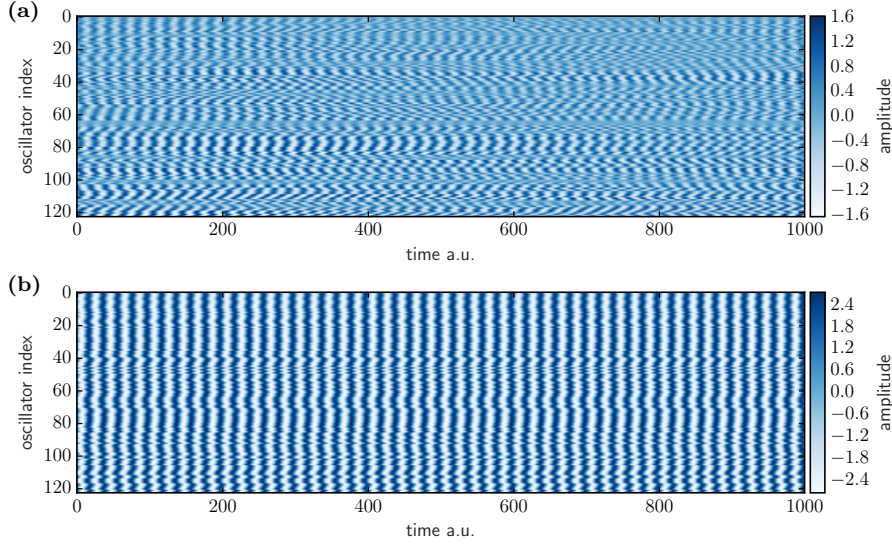


Fig. 28 Oscillators in a hierarchical network. **a:** uncontrolled system; **b:** Pareto-front solution, $u(\dot{\mathbf{x}}) = \sin(\dot{x}_7)$, for forced synchronization.

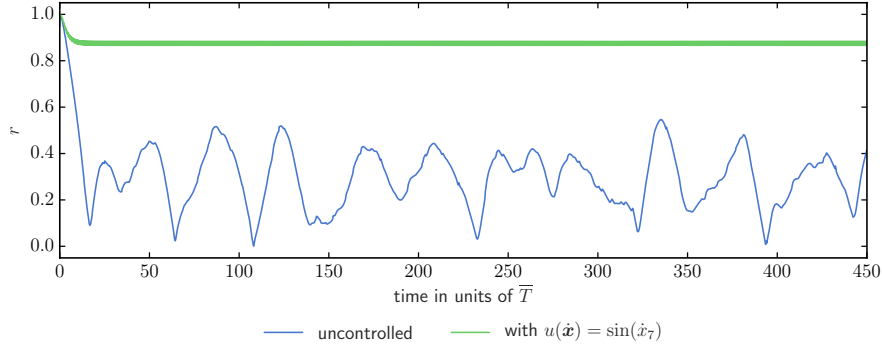


Fig. 29 Oscillators in a hierarchical network. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to Γ_1 (green).

Table 21 Oscillators in a hierarchical network: System setup for forced de-synchronization.

dynamic system		GP	
α, β, c	$0.1, 1, 5.6 \cdot 10^{-2}$	cost functionals	$\exp(\text{std}(\boldsymbol{\Omega}))$
$\bar{\omega}, \Delta\omega$	$\ln(4), 2 \cdot 10^{-2}$		$\text{length}(u)$
ω_i	$\text{linspace}(\bar{\omega} - \Delta\omega, \bar{\omega} + \Delta\omega, 123)$	argument set	$\{\dot{x}_i\}_{i=0, \dots, 11}$
$\mathbf{x}_i(t_0)$	1 ($i = 0, \dots, 122$)	constant set	$\{k\}$
$\dot{\mathbf{x}}_i(t_0)$	0 ($i = 0, \dots, 122$)	seed	8797055239111497159
t_0, t_n	$0, 2000 \frac{2\pi}{\bar{\omega}}$		
n	40000		

global desynchronization in each single oscillator, we need to design the cost function with more care!

Table 22 Oscillators in a hierarchical network: Best solutions for forced de-synchronization.

synchronicity	length	expression	constant
0.722	17	$-(-\dot{x}_3 + (-\dot{x}_8)) + (\dot{x}_{11} + \dot{x}_8 + \sin(\dot{x}_9)) \cdot (-\exp(\dot{x}_0))$	$k = 1$
0.727	16	$-(-\dot{x}_3 + (-\dot{x}_8)) + (\dot{x}_{11} + \dot{x}_5 + \dot{x}_8) \cdot (-\exp(\dot{x}_{11}))$	
0.749	13	$\dot{x}_{11} + \dot{x}_8 + (\dot{x}_{11} + \dot{x}_8 + \dot{x}_9) \cdot (-\exp(\dot{x}_6))$	
0.886	4	$-\exp(\sin(\dot{x}_0))$	
0.886	3	$-\exp(\dot{x}_0)$	
0.997	2	$-\dot{x}_8$	
1.000	1	k	
1.000	1	\dot{x}_4	

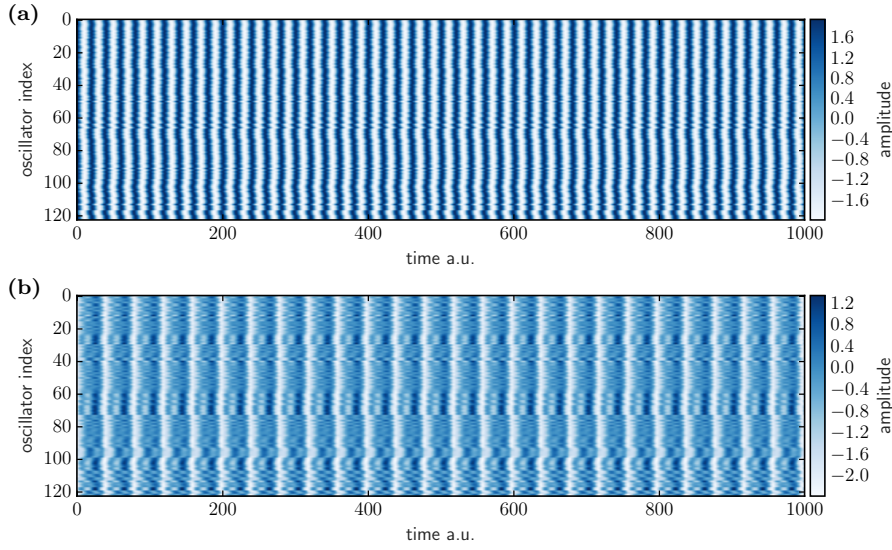


Fig. 30 Oscillators in a hierarchical network. **a**: uncontrolled system; **b**: Pareto-front solution, $u(\dot{\mathbf{x}}) = \dot{x}_3 + \dot{x}_8 - (\dot{x}_{11} + \dot{x}_8 + \sin(\dot{x}_9)) \cdot \exp(\dot{x}_0)$, for forced de-synchronization.

5 Conclusions and Future Work

Our main question, in this work, concerns the control of a synchronization in systems of coupled oscillators. We presented a computational intelligence-based framework for inferring optimal control laws to achieve this goal. A multi-objective genetic programming algorithm with regression-based constant estimation is used to learn the control laws dynamically.

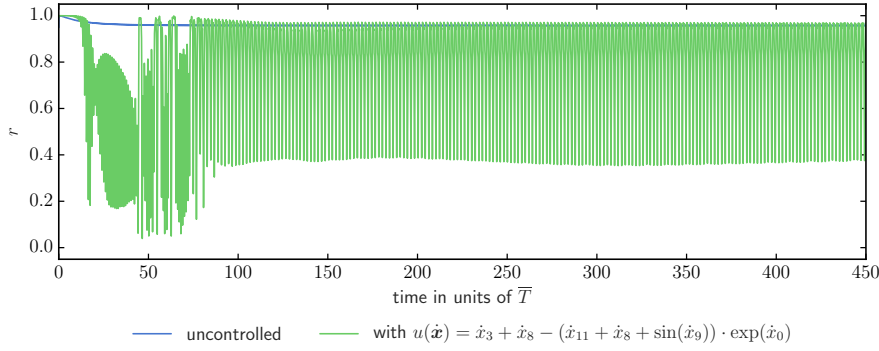


Fig. 31 Oscillators in a hierarchical network. Kuramoto order parameter r for the uncontrolled system (blue) and the system controlled by the best solution, with respect to Γ_1 (green).

We first tested our method on two well-known control problems in dynamical systems: i) drive a damped harmonic oscillator to a limit cycle and a stable one to a fixed point; and ii) drive the Lorenz system from a chaotic state to regular dynamics. We then applied our control approach to dynamical systems composed of networks of coupled oscillators, starting from a system of two coupled van der pol oscillators up to a hierarchical network consisting of a few hundred oscillators. As intermediate steps we evaluated our approach on a ring of oscillators and a torus.

As a result we find terms of different complexity leading to different levels of synchronization control, where synchronization is measured using the Kuramoto parameter. The results clearly demonstrate the ability of GP-based control to bring a desynchronized system to a synchronization state and vice versa. For forced synchronization, in any setup we find simple control laws, suggesting a single oscillator taking over the control and governing the overall dynamics. For forced desynchronization, laws of increasing complexity are found, where the complexity increased with that of the system complexity.

The results in all setups highlight the importance of designing the objective functions appropriately. In the current setup, we simply relied on learning the control laws by minimizing the error with the desired output. We did not specify any symmetry or energy function to be minimized, nor did we restrict the number of oscillators to be controlled. These details appear to be important for using our methods in a real-world application.

Further work will extend current methods in several ways: a subsequent automatic stability analysis would be performed for the numerical experiments. This way one can immediately distinguish stable and useful control dynamics from unstable ones. Second, we aim to look into the design of better objective functions, taking into consideration prior domain knowledge, e.g. in the form of additive symmetry terms. Finally, we plan to integrate methods in our

framework that would search for the optimal sensor (for measurement) and pressure (for actuation) points in the network for a better control.

References

1. N. Gautier, J.L. Aider, T. Duriez, B.R. Noack, M. Segond, M.W.. Abel, J. Fluid Mech. **770**, 242 (2015)
2. E. Ott, C. Grebogi, J.A. Yorke, Phys. Rev. Lett. **64**, 1196 (1990). DOI 10.1103/PhysRevLett.64.1196. URL <http://link.aps.org/doi/10.1103/PhysRevLett.64.1196>
3. G. Chen, X. Yu, *Chaos control: theory and applications*, vol. 292 (Springer Science & Business Media, 2003)
4. H. Haken, *Brain Dynamics: Synchronization and Activity Patterns in Pulse-Coupled Neural Nets with Delays and Noise*. Springer Series in Synergetics (Springer Berlin Heidelberg, 2006). URL <https://books.google.de/books?id=8e1DAAAQBAJ>
5. J.M. Schwalb, C. Hamani, Neurotherapeutics **5**(1), 3 (2008)
6. A. Pikovsky, M. Rosenblum, J. Kurths, *Synchronization: A Universal Concept in Non-linear Sciences*. Cambridge Nonlinear Science Series (Cambridge University Press, 2003)
7. S.H. Strogatz, *Sync: How order emerges from chaos in the universe, nature, and daily life* (Hyperion, 2003)
8. T. Yang, L.O. Chua, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications **44**(10), 976 (1997)
9. Z. Li, X. Cao, N. Ding, IEEE Transactions on Fuzzy Systems **19**(4), 745 (2011)
10. C. Hammond, H. Bergman, P. Brown, Trends in neurosciences **30**(7), 357 (2007)
11. F. Dörfler, M. Chertkov, F. Bullo, Proceedings of the National Academy of Sciences **110**(6), 2005 (2013)
12. D.E. Kirk, *Optimal control theory: an introduction* (Courier Corporation, 2012)
13. J. Nocedal, S. Wright, *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering (Springer New York, 2006). URL <https://books.google.de/books?id=eN1PAAAAAAAJ>
14. S. Sra, S. Nowozin, S.J. Wright, *Optimization for Machine Learning* (MIT Press, Cambridge, USA, 2011)
15. V.M. Becerra, Scholarpedia **3**(1), 5354 (2008). revision 124632
16. D.E. Kirk, *Optimal Control Theory* (Prentice-Hall, 1970)
17. E. Schöll, H.G. Schuster, *Handbook of chaos control* (John Wiley & Sons, 2008)
18. M. Schmidt, H. Lipson, Science **324**(5923), 81 (2009)
19. E.J. Vladislavleva, G.F. Smits, D. Den Hertog, IEEE Transactions on Evolutionary Computation **13**(2), 333 (2009)
20. M. Quade, M. Abel, K. Shafi, R.K. Niven, B.R. Noack, Phys. Rev. E **94**, 012214 (2016). DOI 10.1103/PhysRevE.94.012214. URL <http://link.aps.org/doi/10.1103/PhysRevE.94.012214>
21. V. Parezanovic, L. Cordier, A. Spohn, T. Duriez, B.R. Noack, J.P. Bonnet, M. Segond, M. Abel, S.L. Brunton, J. Fluid Mech. **797**, 247 (2016). DOI 10.1017/jfm.2016.261
22. S.H. Strogatz, D.A. Wiley, M. Girvan, Chaos – An Interdisciplinary Journal of Nonlinear Science **16** (2006). URL <http://dx.doi.org/10.1063/1.2165594>
23. F.L. Lewis, V.L. Syrmos, *Optimal Control* (John Wiley and Sons, 1995)
24. A.E. Bryson (Jr), Y. Ho, *Applied Optimal Control* (Halsted Press, 1975)
25. M. Athans, P.L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications* (Dover Publications, 2006)
26. G.F. Franklin, J.D. Powell, A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 7th edn. (Pearson, 2014)
27. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, 1992)
28. N.L. Cramer, in *Proceedings of an International Conference on Genetic Algorithms and the Applications*, ed. by J.J. Grefenstette (Psychology Press, 1985), pp. 183–187
29. Poli, W.B. Langdon, N.F. McPhee, *A Field Guide to Genetic Programming* (Lulu, 2008)
30. X. Yang, Scholarpedia **6**(8), 11472 (2011). revision 91488

31. S. Luke, *Essentials of Metaheuristics*, 2nd edn. (Lulu, 2013)
32. D. Fudenberg, J. Tirole, *Game theory* (MIT press, 1991)
33. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, in *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature* (Springer-Verlag, 2000), pp. 849–858
34. J. Knowles, D. Corne, in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1 (IEEE, 1999), vol. 1
35. E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm (2001)
36. F.A. Fortin, F.M. De Rainville, M.A. Gardner, M. Parizeau, C. Gagné, *Journal of Machine Learning Research* **13**, 2171 (2012)
37. S. van der Walt, S.C. Colbert, G. Varoquaux, *Computing in Science & Engineering* **13**(2), 22 (2011). DOI 10.1109/IEEESTD.2000.92296
38. E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python (2001). URL <http://www.scipy.org/>. [Online; accessed 2016-07-13]
39. M. Matsumoto, T. Nishimura, *ACM Transactions on Modeling and Computer Simulation* **8**(1), 3 (1998)
40. SymPy Dev Team. SymPy: Python library for symbolic mathematics (2016). URL <http://www.sympy.org>
41. E.N. Lorenz, *Journal of the Atmospheric Sciences* **20**(2), 130 (1963). DOI 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2. URL [http://dx.doi.org/10.1175/1520-0469\(1963\)020<0130:DNF>2.0.CO;2](http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2)
42. E. Bullmore, O. Sporns, *Nature Reviews – Neuroscience* **10**, 186 (2009). DOI 10.1038/nrn2575
43. Y. Kuramoto, in *International Symposium on Mathematical Problems in Theoretical Physics*, vol. 39, ed. by H. Araki (Springer, 1975), vol. 39, p. 420
44. Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence* (Springer-Verlag, 1984)
45. L. Cohen, *Time Frequency Analysis: Theory and Applications*, 1st edn. (Prentice Hall, 1994)